UNIVERSIDAD NACIONAL DE LA PLATA

FACULTAD DE INFORMÁTICA

TESINA DE LICENCIATURA

Título: Herramienta de autoría de feedback háptico para aplicaciones de realidad virtual

Autor: Milton Maciel Loyola

Director: Gustavo Rossi

Codirector: Andrés Rodríguez

Carrera: Licenciatura en Sistemas

Resumen

En este trabajo presentamos una herramienta de autoría háptica desarrollada sobre el motor de videojuegos Unity que permite el prototipado de estímulos táctiles para ser usados en entornos de realidad virtual.
Ante la reciente aparición de cascos de realidad virtual para consumo masivo, detectamos la ausencia de metodologías para integrar patrones hápticos en entornos virtuales, donde la simulación de presencia juega un rol fundamental pero el desarrollo de medios hápticos se encuentra rezagado comparado con sus contrapartes visuales y auditivas.
Para construir esta herramienta, retomamos investigaciones realizadas en los últimos años sobre metodologías para proveer autoría de patrones táctiles, con la premisa de plasmarlo en medios vigentes de creación de entornos virtuales. Con este fin, integramos productos existentes como son Unity, Oculus Rift y Leap Motion con un display háptico compuesto por pequeños motores de vibración controlados por una placa Arduino, los cuales conforman nuestro entorno de ejecución.
La herramienta es concebida como una prueba de concepto sobre cómo integrar háptica en realidad virtual de forma sencilla y accesible para diseñadores, aprovechando el potencial de las tecnologías que se encuentran disponibles actualmente.

Palabras Claves

Háptica, herramienta de autoría, realidad virtual, feedback vibrotáctil, patrón vibrotáctil, diseño de estímulos táctiles.

Trabajos Realizados

En la realización de este trabajo las actividades realizadas fueron, en principio, el relevamiento de los principales trabajos de investigación en herramientas hápticas de los últimos 15 años. La definición de requisitos, alcance y funcionalidades que debía cumplir la herramienta. La puesta en marcha del entorno de ejecución, de los dispositivos de realidad virtual y su integración en la plataforma de desarrollo. El desarrollo de la herramienta, contemplando pruebas, evaluaciones e iteraciones hasta alcanzar su forma final.

Conclusiones

En esta tesina presentamos una herramienta para diseñar estímulos hápticos en Unity que conforma un modelo experimental sobre cómo se articula una herramienta de autoría háptica dentro de un entorno de desarrollo de realidad virtual. Esta herramienta fue utilizada para recrear experiencias básicas de realidad virtual con interfaces vibrotáctiles en las cuales pudimos comprobar su facilidad de uso y la rapidez en el ciclo de diseño, prueba y ejecución de interacciones hápticas.

Trabajos Futuros

Los objetivos a futuro contemplan enriquecer la herramienta con la inclusión de modos de diseño basados en características físicas de los materiales y el desacople de las tecnologías vinculadas actualmente para soportar otras plataformas y dispositivos. Otras metas de estudio relacionadas al rendering háptico son la reducción de las distorsiones de los estímulos percibida por los usuarios.

Fecha de la presentación: JULIO DE 2017

UNIVERSIDAD NACIONAL DE LA PLATA
FACULTAD DE INFORMÁTICA

HERRAMIENTA DE AUTORÍA DE FEEDBACK HÁPTICO PARA APLICACIONES DE REALIDAD VIRTUAL

TESINA DE LICENCIATURA EN SISTEMAS

MILTON MACIEL LOYOLA

DIRECTOR: GUSTAVO ROSSI
CODIRECTOR: ANDRÉS RODRÍGUEZ

LA PLATA, JULIO DE 2017

RESUMEN

En este trabajo presentamos una herramienta de autoría háptica desarrollada sobre el motor de videojuegos Unity que permite el prototipado de estímulos táctiles para ser usados en entornos de realidad virtual.

Ante la reciente aparición de cascos de realidad virtual para consumo masivo, detectamos la ausencia de metodologías para integrar patrones hápticos en entornos virtuales, donde la simulación de presencia juega un rol fundamental pero el desarrollo de medios hápticos se encuentra rezagado comparado con sus contrapartes visuales y auditivas.

Para construir esta herramienta, retomamos investigaciones realizadas en los últimos años sobre metodologías para proveer autoría de patrones táctiles, con la premisa de plasmarlo en medios vigentes de creación de entornos virtuales. Con este fin, integramos productos existentes como son Unity, Oculus Rift y Leap Motion con un display háptico compuesto por pequeños motores de vibración controlados por una placa Arduino, los cuales conforman nuestro entorno de ejecución.

La herramienta es concebida como una prueba de concepto sobre cómo integrar háptica en realidad virtual de forma sencilla y accesible para diseñadores, aprovechando el potencial de las tecnologías que se encuentran disponibles actualmente.

TABLA DE CONTENIDOS

1 INTRODUCCIÓN

En este capítulo realizamos una introducción al trabajo de esta tesina, en donde explicamos las principales motivaciones, identificamos los objetivos que buscamos alcanzar junto con los objetivos secundarios. Al final presentamos una descripción de la organización de este documento, mostrando un resumen de los diferentes capítulos.

1.1 MOTIVACIÓN

Recientemente vimos un resurgimiento de tecnologías de realidad virtual que por primera vez se encuentran presentes en el mercado de consumo masivo y en los hogares. Los sentidos cubiertos por estas tecnologías son el visual y el auditivo, con el sentido táctil aún ausente o integrado de manera accesoria a pesar de ser un elemento esencial en la sensación de presencia. Asimismo, hay muy pocas herramientas para el diseño de estímulos vibrotáctiles, y ninguna en plataformas de desarrollo de realidad virtual.

Si bien se pueden encontrar numerosos proyectos que incluyen el diseño de dispositivos hápticos, sorprendentemente la mayoría de ellos están hechos sin el uso de ninguna herramienta de prototipado que ofrezca algún soporte al diseñador. Sin embargo, es aún difícil desarrollar aplicaciones hápticas y, en consecuencia, es necesario ocultar la complejidad de programar una aplicación háptica o interacciones proporcionando herramientas de prototipado adecuadas.

Actualmente, existen algunas propuestas que ofrecen diferentes alternativas para facilitar a diseñadores sin conocimientos de progra-

mación la creación de prototipos hápticos. Las herramientas orientadas al tacto, usualmente proveen el diseño de estímulos utilizando una GUI para facilitar la edición de curvas, y posteriormente, exportan los estímulos en un formato adecuado para ser empleado por una aplicación externa que los reproduce en respuesta a una serie de eventos predefinidos. Otras herramientas, se basan en una integración de dispositivos de force feedback con herramientas de creación de entornos virtuales, que permiten configurar objetos 3D y dotarlos de háptica utilizando un sistema de colisiones para facilitar la detección de formas y volúmenes de objetos 3D.

En el caso de tecnologías táctiles, las herramientas de autoría existentes presentan una serie de impedimentos. El principal problema es que el diseño de estímulos suele presentarse disociado de la especificación de los eventos que los desencadenan en el entorno virtual. Esto se explica porque los editores no están embebidos dentro de una herramienta de creación de entornos virtuales, sino que son provistas como herramientas independientes. La capacidad de contar con un entorno de creación de mundos virtuales que incluya el diseño háptico permitiría al diseñador contar con un flujo de trabajo integral, desde la especificación de los estímulos hápticos, hasta la configuración de los eventos que desencadenan su reproducción. Del mismo modo, se presenta conveniente tener los estímulos hápticos asociados al objeto como una parte más de si mismo. El otro inconveniente de las herramientas actuales es que son soluciones innovadoras sin una fuerte base de referencia y, la naturaleza de las mismas, es que presentan interfaces de usuario gráficas que precisan conocimientos previos en el área para ser utilizadas. Del mismo modo, se vuelve difícil evaluarlas en términos de usabilidad.

Este contexto es propicio para la exploración de alternativas que provean herramientas de autoría háptica que estén en consonancia con las metodologías vigentes sobre cómo desarrollar experiencias de realidad virtual. Es preciso un estudio de la evolución de las herramientas hápticas que han sido empleados hasta ahora y evaluar como

los diseñadores desarrollan aplicaciones de realidad virtual en la actualidad.

En este trabajo presentamos una herramienta de autoría para diseñar y testear patrones vibrotáctiles en entornos de realidad virtual. Los patrones táctiles permiten acentuar notablemente la experiencia en realidad virtual cuando los medios visuales y auditivos resultan insuficientes. La herramienta, que funciona como una integración del motor de videojuegos Unity3D, permite a diseñadores crear estímulos hápticos usando representaciones gráficas en un entorno familiar para el desarrollo de experiencias de realidad virtual.

El soporte a la reproducción táctil en medios de realidad virtual se brinda mediante la integración con un display háptico compuesto por pequeños motores de vibración controlados por una placa Arduino y el sensor de reconocimiento de gestos Leap Motion.

1.2 Objetivos

Concebir, especificar y desarrollar una herramienta de autoría de interacción háptica por estimulación vibrotáctil dirigida a diseñadores. La herramienta facilitará la creación y manipulación de patrones de interacción hápticos dentro de un mundo virtual. El editor debe ser manipulable por un diseñador de interacción y proveer resultados que se puedan probar directamente con facilidad y sin necesidad de escribir código. Las herramientas deben ser accesibles y tener alcance en su empleo, con lo cual se realizan como extensión de un motor de videojuegos gratuito y se utiliza hardware de acceso comercial o que se pueda construir económicamente.

Los objetivos secundarios son:

- Identificar los criterios de usabilidad deseables en la GUI.

- Desarrollar un plugin para Unity3D que implemente la interfaz con el dispositivo háptico.
- Crear un mecanismo para asociar la disposición física de los sensores vibrotáctiles del guante con el esqueleto de mano utilizado por Leap Motion para sensar la posición.
- Desarrollar el editor visual de patrones hápticos.
- Proponer mecanismos de composición de objetos del mundo virtual para dotarlos de funciones hápticas.
- Realizar experimentos en forma entornos de ejecución sobre los conceptos provistos por la herramienta para la evaluación de las herramientas.

1.3 Estructura de la tesina

- El capítulo *Marco teórico* brinda descripciones de contexto de los elementos conceptuales fundamentales en que se basa esta tesina.
- En *Trabajos relacionados* hay una revisión de estudios anteriores vinculados con este sobre patrones vibrotáctiles, editores de háptica e integraciones de háptica en entornos de desarrollo de realidad virtual.
- En *Requerimientos* se delinean las funcionalidades y objetivos que se procuran elementales para el desarrollo de la herramienta propuesta en este trabajo.
- El capítulo *Diseño conceptual* describe la arquitectura de los elementos de hardware y software que componen la herramienta.
- El capítulo *Implementación* contiene los detalles de realización de la herramienta.

- En *Evaluación* hay una descripción de las pruebas de concepto realizadas para verificar el cumplimiento de los objetivos propuestos, junto con las preliminares del empleo de la herramienta.
- En *Conclusiones,* hay un resumen de los aspectos contribuidos y se delinea una proyección de cómo podría evolucionar el trabajo.

2 MARCO TEÓRICO

En este capítulo describimos los principales conceptos teóricos en que se funda este trabajo, los cuales utilizamos como preliminares para la comprensión del resto del documento. Se definen las terminologías empleadas en el resto de los capítulos, se exponen las principales definiciones de los conceptos teóricos y por últimos se ofrece un contexto de las tecnologías involucradas.

2.1 HÁPTICA

Háptica, que en general refiere al sentido del tacto, juega un rol esencial no solo en nuestra construcción sensorial de la disposición espacial, sino también en la habilidad humana para manipular objetos con nuestras manos. Según Lederman y Klatzky [1], la visión y audición son reconocidos por proveer con precisión información espacial y temporal, respectivamente, mientras que los sistemas hápticos son espacialmente efectivos procesando las características materiales de superficies y objetos. Además, este sentido refuerza otros canales e incrementa la percepción de presencia en entornos virtuales.

El pasaje de interfaces de usuario tradicionales, compuestas por pantallas, mouse y teclado, a las modernas como las pantallas táctiles presentes en teléfonos móviles, resultó en la perdida de la percepción de las teclas que permitían su uso eficiente sin ayudas visuales. En el teclado, no solo la forma y disposición de las teclas son fácilmente percibidas por el sentido del tacto del usuario, sino que también se reconoce fácilmente su activación. En cambio, en una pantalla táctil, los canales visuales necesitan hacer un soporte activo a la interacción

del usuario. En la industria se ha reemplazado parcialmente esta ausencia con el feedback vibrotáctil, el cual está presente en prácticamente todos los dispositivos móviles. Los controles de consolas de videojuegos también se benefician de este feedback, a diferencia de que el objetivo es enriquecer la experiencia de usuario.

Recientemente, las tecnologías de dispositivos de interacción han dado el salto más allá de la pantalla, con sensores de profundidad y detección de movimiento. Por ejemplo, Microsoft produjo el sensor Kinect[1] para su consola de videojuegos Xbox 360. A su vez, el sensor de movimiento Leap Motion[2], el dispositivo que empleamos en este trabajo, fue desarrollado inicialmente por David Holz mientras estudiaba para un doctorado en matemáticas. Holz comenzó a desarrollar el software en 2008 en la Universidad de Carolina del Norte, donde frustrado por las limitaciones del mouse y teclado estuvo cinco años desarrollando las cámaras para el controlador [2].

En estos casos, la ausencia de feedback háptico es aún más notoria ya que los gestos ocurren en el aire. El problema no es nuevo, de hecho, estuvo presente en realidad virtual durante muchos años dado que el desafío es proveer sensaciones realistas al usuario. Muchos entornos de realidad virtual tienen gráficos asombrosos y vienen equipados con sonido de alta fidelidad, mientras que la tecnología háptica está claramente atrasada. Sin embargo, tener la posibilidad de tocar, sentir y manipular objetos, además de ver y oírlos, es esencial para satisfacer los objetivos de la realidad virtual.

[1] http://www.xbox.com/xbox-one/accessories/kinect
[2] http://www.leapmotion.com

Figura 1: Dispositivos sensores de movimiento. De arriba hacia abajo: Kinect for Xbox 360, Asus Xtion, Leap Motion.

2.2 Realidad virtual

Realidad virtual refiere comúnmente a tecnologías de computadora que utilizan software para generar imágenes realistas, sonidos y otras sensaciones para replicar un entorno real, o crear un entorno imaginario, con lo cual simulan la presencia física del usuario en el entorno. Permitiendo al usuario interactuar con este espacio y todos los objetos que contiene utilizando pantallas y otros sensores. Una persona utilizando dispositivos de realidad virtual es normalmente capaz de mirar alrededor en el mundo artificial, moverse por el entorno e interactuar con personajes u objetos que aparecen en la pantalla o los gafas. La realidad virtual recrea artificialmente experiencias sensoriales, que pueden incluir la vista, el tacto, la audición, y menos comúnmente, el olfato.

Inmersión

La función que cumple la realidad virtual cuando es utilizada en entretenimiento es aumentar la inmersión en la experiencia de usuario.

La inmersión dentro de la realidad virtual trata de la percepción de estar presente físicamente en un mundo ficticio. Una forma de ver la experiencia de inmersión virtual podría ser como el conjunto de técnicas para persuadir al usuario con información sensorial para recrear la presencia en un entorno ficticio.

Actualidad

En los últimos años se vio un resurgimiento de las tecnologías de realidad virtual que tuvo como origen la incubación del casco de realidad virtual Oculus Rift[3] y tecnologías de reconocimiento de gestos como Kinect y Leap Motion las cuales tienen como propósito la llegada de estos paradigmas de interacción con computadora al público masivo. El dispositivo Leap Motion es un sensor que soporta como entrada los movimientos de las manos y sus dedos, en analogía a un mouse, pero no requiere contacto de la mano o tacto. A partir de 2016, el dispositivo fue adaptado para soportar detección de gestos en realidad virtual.

En la actualidad, en la industria de la realidad virtual, se están enfatizando los esfuerzos para lograr la estabilidad de los sistemas de visión (que aún provocan trastornos de náuseas por simulación) para una experiencia satisfactoria del usuario. Dentro de los demás elementos de interacción de los cuales se espera haya una evolución, propiciada por la inserción de los cascos en el mercado masivo, están el reconocimiento de gestos y los sistemas hápticos. Si bien se considera el clásico gamepad como dispositivo de entrada predilecto para usar los cascos de realidad virtual, ya han surgido propuestas más inmersivas como la adaptación del Leap Motion para su uso acoplado al Oculus Rift [3] o al HTC Vive [4].

[3] http://www.oculus.com/rift/

La integración del sensor Leap Motion a los cascos de realidad virtual dio origen al lanzamiento de algunos videojuegos con el reconocimiento de gestos como medio de entrada. También se produjeron una gran cantidad de aplicaciones o juegos como pruebas de conceptos realizados de manera experimental. Algunos ejemplos son, en el campo de las aplicaciones industriales, una aplicación para el entrenamiento en intervenciones quirúrgicas [5], la exploración de modelos digitales de construcciones en realidad virtual [6] y un proyecto para interactuar con un instrumento virtual conceptual [7].

En el campo de la háptica, en cambio, la irrupción de dispositivos y herramientas apropiadas que puedan ser adoptadas al público masivo es aún un tema pendiente. Si bien se están incubando varios proyectos orientados a suplir esta faltante.

Figura 2: Integración de Leap Motion con montura para Oculus Rfit (izquierda) y HTC Vive (derecha).

2.3 TECNOLOGÍAS DE RETROALIMENTACIÓN HÁPTICA

Por retroalimentación háptica referimos al empleo del canal de comunicación táctil en interfaces de usuario hombre-máquina. El término feedback háptico dentro de los sistemas sensoriales incluye el tacto como así también la kinestesia [8]. Siendo el tacto asociado al

subsistema sensorial cutáneo y el sentido kinestésico que refiere a la sensación de movimiento o tensión en músculos, tendones y articulaciones. En tecnologías hápticas típicamente se utilizan motores vibradores para el sentido táctil y mecanismos de aplicación de fuerzas para el sentido kinestésico, aunque no son las únicas.

En este trabajo utilizaremos el término feedback vibrotáctil para referir a la transmisión de estímulos utilizando medios estrictamente táctiles. Por otro lado, con el término force feedback haremos referencia a estímulos creados a partir de fuerzas que actúan en el sentido kinestésico. De este modo, podemos catalogar a ejemplos prácticos, los guantes con motores vibradores como un medio de feedback vibrotáctil y, por otro lado, dispositivos como el Phantom Omni [9],una especie de bolígrafo motorizado que aplica fuerzas en las manos, como un medio de force feedback.

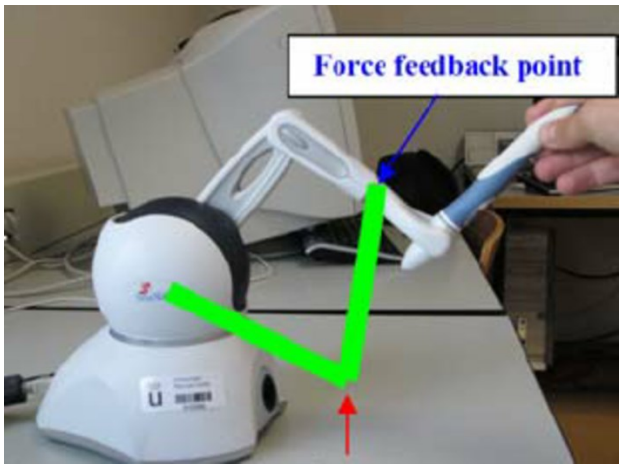


Figura 3: Ejemplo de aplicación de force feedback en el dispositivo Phantom Omni.

2.4 Estímulos hápticos

En este trabajo utilizamos frecuentemente como terminología las denominaciones estímulo háptico, patrón háptico e ícono háptico para referir a conceptos muy similares, los cuales vamos a diferenciar a continuación.

En principio, con estímulo háptico referimos a un símbolo representado de forma táctil. El alcance de esta denomiación es para referir tanto a la representación del estímulo táctil en la herramienta de autoría como así también al conjunto de señales reproducido por un display táctil y percibido por un usuario. Luego el término ícono háptico es utilizado asiduamente en referencias bibliográficas de años anteriores para hacer referencia principalmente a las representaciones de los estímulos hápticos. El término ícono háptico continúa la idea de representació simbólica y plantea una analogía con los íconos gráficos de uso cotidiano. Por último, con patrón háptico referimos a la representación del estímulo en la herramienta de autoría, la cual puede ser editada, guardada y reproducida. En particular, en este trabajo los patrones hápticos son un conjunto de curvas que representan cada una la fuerza de una señal vibrotáctil emitida sobre una parte de la mano del usuario.

2.5 Transmisión de estímulos táctiles

La transmisión de estímulos utilizando dispositivos de tecnología vibrotáctil se realiza en una serie de etapas. El proceso de generar una salida táctil para el usuario mediante software se denomina rendering háptico. En este paso, los estímulos diseñados son convertidos a señales analógicas que activan un motor vibrotáctil. Un estímulo vibrotáctil es generado desde el motor y transmitido a la superficie del

envoltorio donde está embebido el actuador. Por último, tiene lugar la percepción, en donde las vibraciones son sentidas a través de la piel del usuario en contacto con la envoltura [10].

2.6 Displays hápticos

La transmisión de estímulos táctiles se logra mediante displays táctiles, es decir, dispositivos que en contacto con la piel son capaces de lograr sensaciones de mayor o menos complejidad [11]. Las características principales que diferencian a estos displays entre si son su resolución espacial, el tiempo de respuesta y la fuerza o intensidad que pueden ejercer. Siendo el factor determinante para la efectividad de la transmisión de sensaciones la tecnología empleada. Algunas tecnologías empleadas actualmente son la neumática, electromagnética, electrostática, piezoeléctrica, aleaciones con efecto térmico, entre otras. En este trabajo utilizamos displays lo suficientemente ligeros y pequeños para adaptarse a un guante y nos referimos a ellos como motores o actuadores indistintamente.

Desarrollos previos

En los sucesivos esfuerzos por desarrollar esquemas de interacción basados en háptica, varios investigadores estudiaron diferentes formas de proveer sensaciones realistas, y diferentes compañías crearon dispositivos complejos como Phantom Omni o CyberGrasp[4]. Estos productos comerciales, enfrentan las limitaciones de tener un espacio de trabajo reducido y altos costos de adquisición, lo que los ha

[4] http://www.cyberglovesystems.com/cybergrasp/

limitado a un reducido campo de aplicaciones. Esta falta de soluciones táctiles de propósito general ha llevado a varios investigadores a producir sus propios dispositivos.

Figura 4: Dispositivo de force feedback CyberGrasp de CyberGlove Systems Inc.

Los prototipos de dispositivos hápticos construidos recientemente vienen propiciados por placas del tipo Single Board Computer o SBC y actuadores táctiles de bajo costo como los actuadores lineales resonantes (LRA) o motores vibradores ERM. Las placas SBC como Arduino o Raspberry Pi consisten en computadoras completas, con microprocesador, memoria y entrada/salida, ensamblados en un solo circuito y disponibles a bajo coste. Los actuadores táctiles tienen la característica de ser lo suficientemente ligeros y pequeños para adaptarse al cuerpo del usuario y así poder formar un guante.

Un ejemplo de soluciones construidas con estas tecnologías en el ámbito de investigaciones académicas es el kit de feedback háptico llamado Vitaki [11]. El cual consiste en un controlador para un dispositivo que permite una fácil colocación de actuadores táctiles formando una salida háptica adaptable a diferentes escenarios. El controlador soporta hasta 32 actuadores que, por ser de propósito general, pueden ser distribuidos en cualquier parte del cuerpo.

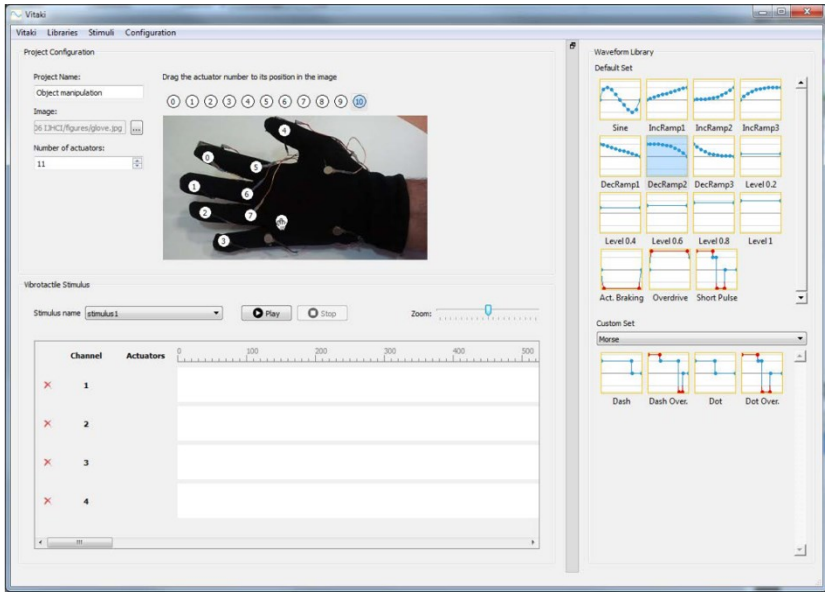


Figura 5: Editor de la herramienta de autoría háptica Vitaki.

3 TRABAJOS RELACIONADOS

En este capítulo presentamos un desglose de los principales trabajos publicados en los últimos quince años en materia de herramientas de autoría háptica junto con breves reseñas del software comercial de misma finalidad. Los trabajos son divididos según el sistema sensorial objetivo, según las metáforas empleadas para representar estímulos y según la plataforma de desarrollo.

3.1 HERRAMIENTAS DE AUTORÍA PARA FEEDBACK VIBROTÁCTIL

El campo estudio de interfaces hápticas tiene menos desarrollo que sus contrapartes visuales o auditivas, sin embargo, se presenta como un área de estudio en creciente crecimiento. Muchas investigaciones refieren al empleo de háptica para mejorar entrenamientos, como simulaciones de cirugías médicas, o para reforzar la sensación de presencia en casos de teleoperaciones[5]. En este relevamiento, nos vamos a centrar en los métodos empleados para mejorar la sensación de presencia con háptica utilizando dispositivos vibrotáctiles, con el objetivo final de ser empleado en realidad virtual

Los trabajos de investigación sobre herramientas de autoría háptica que se realizaron podemos clasificarlos en base a diferentes objetivos que se propusieron y los contextos de ejecución que definieron. Una distinción inicial es que algunos editores fueron diseñados para trabajar sobre dispositivos de force feedback mientras que otros fueron pensados para feedback vibrotáctil, nosotros vamos a centrarnos en estos últimos. Otra distinción importante es en el paradigma grá-

[5] Teleoperación es un término comúnmente utilizado para referir a la realización de operaciones sobre maquinas a la distancia

fico que emplearon para editar patrones, algunos desarrollos se basaron en edición de curvas y otros en el empleo de metáforas. Por último, cabe mencionar que, si bien la mayoría de estos editores fueron presentados por investigadores, unos pocos pertenecen a propuestas comerciales, los cuales serán relevados con menos rigor por la falta de divulgación de los conceptos metodológicos empleados.

En este capítulo vamos a repasar los resultados obtenidos de las investigaciones previas, de forma de continuar las implementaciones que reúnen los requisitos que propusimos.

3.1.1 Por representación directa

Algunos editores que han sido desarrollados están basados en una representación directa de las señales como curvas editables. Esto surge por analogía con los editores de sonido, ya que las curvas representando estímulos hápticos pueden ser editados de modo similar a las curvas de audio. El trabajo de íconos auditivos iniciado por Gaver [12] ayudó a proveer un punto de partida inicial para el desarrollo de íconos hápticos. Las representaciones directas, basadas en curvas, se caracterizan por permitir un control refinado de los resultados a través de la manipulación directa de las señales. El requisito para su utilización por los usuarios es que tengan conocimientos básicos de curvas y sus principios matemáticos subyacentes.

Hapticon Editor

Uno de los primeros editores hápticos que tomamos de arquetipo por la inclusión de varias de las abstracciones de diseño presentes hoy en día es Hapticon Editor [13]. El editor permite la creación y edición de patrones hápticos para un display de feedback de un grado de libertad. El dispositivo empleado para probar los patrones consistía en

una perilla giratoria que utiliza fuerzas para transmitir el estímulo. La fuerza o estímulo resultante es un torque o fuerza de rotación. Si bien la perilla representa un perfil de force feedback, el editor podía ser adaptado para permitir el diseño de patrones vibrotáctiles, lo cual posibilitó que los desarrollos que la siguieron se basaran en los mismos principios.

En esta investigación y algunas subsiguientes que se basaron en ella, la terminología para referir a los patrones hápticos, o estímulos hápticos, era la denominación de ícono háptico. Bajo esta metáfora establecían una analogía con los íconos gráficos o auditivos en los que el significado del estímulo es fácilmente reconocible por el usuario. En el presente trabajo utilizamos el termino estímulo háptico para referir a lo mismo.

Un patrón podía ser creado mediante la concatenación de curvas simples en la que la longitud, frecuencia y amplitud pueden ser especificados para cada curva y su forma podía ser modificada a través de puntos de control. La curva representa una función que puede designar, o la cantidad de fuerza aplicada según el tiempo, o la fuerza de resistencia de la perilla según la posición de esta. Según este criterio el empleo de las curvas al reproducirlas podía designar, respectivamente, o un estímulo de force feedback aplicado por única vez, o una descripción espacial describiendo texturas. Esta analogía de las distintas representaciones de las curvas las empleamos en nuestro desarrollo para sustentar diferentes expresiones basadas en una misma herramienta de diseño.

HERRAMIENTA DE PROTOTIPADO DE SWINDELLS ET AL.

El trabajo presentado por Swindells *et al.* [14] es de mayor relevancia por la introducción de una descripción de los requisitos claves que las herramientas deberían tener para diseñar comportamientos há-

pticos. Usando el ejemplo de herramientas de prototipado anteriores, sintetizan atributos de diseño de alto nivel que deben estar presentes. También introducen una herramienta de prototipado de íconos hápticos donde integran las lecciones aprendidas de sus desarrollos anteriores y de pruebas con distintos dispositivos.

Las tres características principales que proponen para crear íconos hápticos son listadas a continuación. Estos requisitos son adoptados en mayor medida por trabajos posteriores en interfaces para diseño de estímulos.

- Un editor de curvas, que representa la magnitud de la señal háptica en espacio o tiempo.
- Una librería de íconos, una colección de elementos representando efectos hápticos básicos.
- Un panel de mosaicos, el cual permita combinar íconos básicos en otros más sofisticados.

Con respecto a las limitaciones psicofísicas de los usuarios y basados en el trabajo de [15], donde se realiza la exploración de umbrales hápticos de una persona utilizando el dedo índice, concluyen que dado la incertidumbre de las limitaciones perceptuales, es más conveniente proveer un control de volumen de las señales hápticas que permita ajustarse mejor a la sensibilidad de cada usuario.

POSVIBEDITOR

En el campo vibrotáctil, el editor posVibEditor [16] se caracterizó por promover técnicas para la portabilidad de patrones hápticos, los cuales pueden ser usados en múltiples tecnologías de motores vibrotáctiles.

En línea con las recomendaciones para editores de autoría háptica publicado por Swindells *et al.* [14], la herramienta provee un editor de patrones en el que el estímulo puede ser diseñado moviendo puntos de control superpuestos a la curva, también un widget de gestión

de patrones diseñados previamente para que puedan ser reutilizados, y una línea de tiempo con múltiples canales donde se pueden combinar de manera intercalada patrones previamente diseñados. Por último, las técnicas de portabilidad de patrones que caracterizan este trabajo consisten en mecanismos de exportación de los patrones a un formato XML con el propósito de mejorar la reusabilidad y extensibilidad de los mismos.

VITAKI GUI

Introducida en el *toolkit* vibrotáctil de Martinez Muñoz [11], la herramienta fue diseñada para asistir al usuario en el diseño y testeo de estímulos vibrotáctiles utilizados con el controlador electrónico que acompaña el *toolkit*.

Basada en la misma metáfora que posVibEditor, la aplicación usa una línea de tiempo de múltiples canales para componer estímulos asociados a uno o más actuadores. Está herramienta agrega la posibilidad de incluir las técnicas de *overdrive* y *breaking* en el diseño. Los patrones diseñados son almacenados y pueden ser reproducidos por una aplicación externa mediante una API.

3.1.2 POR REPRESENTACIÓN INDIRECTA

Otras propuestas que reunimos en esta sección, introducen el uso de metáforas musicales, señales vibrotáctiles simplificadas o están basadas en demostraciones. Estos conjuntos de interfaces han estado orientados a simplificar la tarea de creación de los estímulos hápticos para usuarios no expertos. El mecanismo empleado consiste en desacoplar la composición de patrones vibrotáctiles del proceso de edición a nivel de señal de las características de vibración.

VibScoreEditor

En el editor VibScoreEditor [17] introducen el uso de partituras musicales para representar señales hápticas. La metáfora de la partitura vibrotáctil es un diseño adaptado de las partituras de piano y las tablaturas de guitarra. La representación del estímulo está dada por: localización vertical de las líneas del pentagrama significando la frecuencia de vibración, la forma de la nota representando la duración y un entero dentro de la nota que representa la fuerza. Los resultados a través de las evaluaciones de la herramienta concluyen que el rendimiento de la creación de estímulos mejora comparado con la edición de curvas, aunque dificulta su uso por usuarios no acostumbrados a notación musical y se torna complejo con la inclusión de múltiples actuadores.

Tactile Editor

En TactileEditor [18] usan otra representación gráfica musical tomando inspiración de editores de música y video y sus interfaces de líneas de tiempo. El desarrollo consiste en un editor táctil con una interfaz gráfica basada en pistas orientada a usuarios no expertos. Estos pueden crear patrones hápticos sencillamente mediante la activación y desactivación de motores por determinadas duraciones e intensidades. Esta metáfora de activado/desactivado es sencilla de comprender y no requiere ningún conocimiento previo.

TactiPEd

En TactiPEd [19] presentan un editor para el prototipado sencillo de patrones hápticos basado en la metáfora gráfica del esquema espacial del dispositivo. El cual es utilizado para la configuración de las características principales del estímulo, el cual incluye la amplitud,

frecuencia y duración de secuencias táctiles en múltiples actuadores. Por último, este editor incluye las funcionalidades de exportación de estímulos al formato XML, como así también la reproducción y grabación de los mismos.

HERRAMIENTAS DE AUTORÍA BASADAS EN DEMOSTRACIÓN

Con el método de autoría basado en demostración, un patrón vibrotáctil puede ser generado basado en movimientos del usuario, de este modo se pueden crear patrones fácilmente utilizando gestos.

En la tesis de Hong, K. [20] investigó el mapeo del tacto del usuario a parámetros de vibración para facilitar el diseño de señales vibrotácticas. En esta herramienta, la duración del movimiento, la presión del tacto y la coordenada vertical de la posición del tacto son mapeados a duración, amplitud y frecuencia respectivamente. Esto constituye una forma interactiva y divertida de crear patrones de acuerdo a los usuarios. Sin embargo, debido a la baja precisión de las pantallas táctiles, los gestos son difíciles de controlar y es difícil configurar valores con precisión. Aún más, una vez grabado un patrón, es también difícil editar el patrón ya que requiere volver a grabar el gesto.

En el desarrollo de Cuartielles, Göransson y Olsson de crear herramientas de autoría de patrones hápticos, varios prototipos fueron presentados, donde finalmente proponen un editor donde los patrones hápticos pueden ser creados tocando representaciones iconográficas del cuerpo en una interfaz táctil [21]. Sus editores anteriores usaban líneas de tiempo para controlar cada actuador pero las pruebas realizadas mostraban que la edición de la línea de tiempo para cada actuador consumía mucho tiempo, particularmente en el diseño de patrones hápticos de gran resolución[6].

[6] La resolución en vibrotáctil refiere a la cantidad de actuadores que se usan.

HERRAMIENTAS DE AUTORÍA BASADAS EN SONIDO

Del lado comercial, la compañía Immersion desarrolló Haptic Studio (anteriormente conocido como Motiv Studio) [22], un editor táctil para plataformas móviles. Provee plantillas de efectos hápticos y una interfaz de línea de tiempo para su combinación en patrones hápticos complejos. También provee la generación automática de patrones a partir de archivos de sonido en formato MIDI. Este editor es la única aplicación comercial disponible para diseñar feedback háptico en dispositivos hápticos hogareños. Actualmente la aplicación se está porteando a una plataforma web orientada al diseño háptico para videos comerciales y aplicaciones móviles.

Otra aproximación radicalmente diferente de hacer autoría de contenido háptico es presentada en Techtile toolkit [23] el cual está dirigido a ser una herramienta de prototipado para diseñar háptica de una manera sencilla. Está basada en el mapeo directo de las señales de audio a vibraciones. A través de un micrófono, actuadores de bobina de voz y un amplificador de señal, los usuarios pueden grabar sensaciones utilizando audio y transfiriéndolo a sensaciones hápticas. La experiencia inicial de diseño háptico es mejorada, pero los ajustes finos requieren un editor de audio, los cuales no son particularmente intuitivos.

3.1.3 CONCLUSIONES

En este capítulo varios editores fueron relevados, la mayoría corresponden al período de los últimos quince años e identificamos solo una propuesta comercial para diseñadores. Las interfaces gráficas han sido evaluadas en las investigaciones usando métodos no tradicionales por tratarse de soluciones novedosas, de las cuales solo algunos componentes de interacción han probado su eficacia a través

de diversos estudios que las retomaron sucesivamente. De otros componentes, solo ha prevalecido el carácter exploratorio de las propuestas. Por estas razones, la etapa en que se encuentran estos desarrollos pertenece todavía al período de incubación de estas tecnologías.

Podemos identificar a las herramientas basadas en edición de curvas como las más robustas. Estas han sido la primera aproximación a la edición de patrones hápticos y sobrevivieron a sucesivas iteraciones, mediante las cuales se pudo comprobar su efectividad para representar patrones y el predominio en la capacidad de realizar ajustes finos en ellos. Los métodos de diseño basados en metáforas, si bien constituyen alternativas interesantes, resultan menos convenientes que los editores de curvas para nuestros requisitos propuestos.

Las evaluaciones relevadas de las herramientas arrojan que los paradigmas de metáforas son más convenientes para usuarios finales, con una aproximación más simple. En cambio, los editores con paradigma en curvas, han sido identificados como más idóneos para ser empleados por diseñadores ya que permiten un mayor control del diseño del estímulo. Lo cual se ve exacerbado cuando se utilizan múltiples actuadores.

3.2 Extensiones en plataformas de realidad virtual

Integración de herramienta de autoría háptica en Blender

Blender es un programa de código libre multiplataforma dedicado a la creación de gráficos tridimensionales lanzado inicialmente en 1995. Su popularidad reside en su utilización en muchos animadores independientes y pequeños estudios, debido a factores como su facilidad de uso y el acceso gratuito.

La herramienta de autoría háptica HAMLAT [24] permite a usuarios ampliar con propiedades hápticas el modelado gráfico de objetos virtuales 3D en Blender. La herramienta permite a un modelador de gráficos 3D o un usuario con conocimientos básicos en este ámbito dotar a los objetos virtuales de propiedades hápticas sin tener que programar. La descripción de los modelos hápticos creados por la herramienta está basada en el formato HAML que es un estándar sobre XML diseñado para proveer una descripción de modelos hápticos con neutralidad de tecnología.

La implementación consiste en una extensión de la plataforma Blender construida en base a tres componentes: la herramienta de autoría HAMLAT; el motor de HAML y el reproductor de HAML. La herramienta de autoría es el componente central que extiende la interfaz gráfica de Blender para permitir crear e importar objetos virtuales, activar o desactivar interacciones hápticas, y asignar propiedades hápticas a los objetos seleccionados. El motor de HAML es responsable de generar el archivo HAML que describe todo el entorno virtual y a través del cual el mismo entorno puede ser reconstruido. El reproductor de HAML o renderer es responsable de parsear el archivo HAML y reproducirlo con los recursos disponibles, mediante la invocación de los sistemas de rendering gráficos y hápticos a través de sus APIs.

El aporte de este trabajo consistió en demostrar la utilidad de unificar el modelado gráfico con el diseño háptico aprovechando que muchos estímulos táctiles y kinestésicas son desencadenadas debido a la interacción con objetos virtuales, los cuales son descriptos basados en la geometría de las mallas poligonales. Esta unificación de paradigmas dentro de un editor como Blender simplifica drásticamente el diseño háptico. Los aspectos del diseño háptico que quedan fuera de esta herramienta son el diseño de estímulos que no están representados por la interacción con figuras geométricas. Por esas limitaciones este modelo de herramienta resulta idóneo al ser utilizado en

dispositivos de force feedback y resulta en algunas carencias con los dispositivos vibrotáctiles.

Integración de Phantom Omni en Unity

Phantom Omni es un dispositivo háptico comercial de forcé feedback provisto por Sensable Technologies. De toda la gama de productos hápticos ofrecidos por la compañía, el Phantom Omni está dirigido a consumidores finales. Aun así, el precio es muy elevado para la mayoría de los usuarios, que en 2003 era de 2000 dólares y ahora cuesta aproximadamente 800 dólares. El dispositivo ofrece seis grados de libertad lo cual permite al usuario tocar y manipular objetos en un espacio tridimensional con gran precisión. La forma de interactuar del usuario con el dispositivo es sujetando un accesorio con forma de bolígrafo y de ese modo controlar un cursor en pantalla perteneciente al mundo virtual. El bolígrafo va unido a una base por brazos mecánicos que actúan en conjunción para transmitir las fuerzas al usuario que sostiene el bolígrafo. Las sensaciones que se pueden trasmitir son las características asignadas a distintas geometrías como puede ser su masa (sensación de peso), la fricción, viscosidad y la deformación elástica, relativos a un punto en un espacio tridimensional virtual. También se puede utilizar para simular sensaciones hápticas como las percibidas durante la manipulación de una herramienta portable.

El dispositivo Phantom Omni nunca tuvo soporte oficial de ningún motor de videojuegos. Sin embargo, dado que durante años se ha constituido como un dispositivo robusto de háptica, esto ha motivado a que se hayan realizado extensiones en Unity para utilizarlo en el ámbito de los videojuegos y la realidad virtual. Si bien los mecanismos de force feedback que utilizan dispositivos como el Phantom Omni no son propósito primordial de nuestro proyecto, guardan muchas similitudes con el funcionamiento de los dispositivos vibrotác-

tiles. Es por eso que las extensiones de Unity para Phantom Omni representan un antecedente para nuestra herramienta. A continuación, vamos a describir dos investigaciones independientes entre si que han realizado la integración de Phantom Omni en Unity.

El primer antecedente lo encontramos en la integración del dispositivo Phantom Omni dentro del entorno Unity llevado a cabo en 2006 por Fyans C. y McAllister G. [25] trabajo en el cual se proveyeron un conjunto de scripts de propósito general para dotar una escena 3D con funciones hápticas.

La integración consistió en la construcción de un plugin en C++ para Unity que conectaba el dispositivo a la plataforma de desarrollo y en un conjunto de scripts en C# con tareas básicas de interacciones hápticas implementadas con el propósito de servir de ejemplos y facilitar la curva de aprendizaje de desarrollo. Es importante aclarar que al momento en que se realizó esta integración, Unity funcionaba únicamente en el sistema operativo MacOS y el sistema de plugins de Unity era diferente al empleado actualmente, por lo tanto el plugin consistía en una librería para MacOS empaquetada en un archivo .bundle. La herramienta de desarrollo provista por Sensable era la Phantom Omni API que consistía en dos librerías principales; HDAPI, usada para recolectar información precisa de entrada del dispositivo y enviar representaciones de fuerzas de manera directa al dispositivo háptico; y HLAPI, dirigida a construir entornos hápticos en 3D y generar efectos hápticos limitados como fricción o gravedad. El plugin desarrollado cumplía la función de implementar en un .bundle los métodos de las librerías de Sensable para ser llamados directamente desde Unity y encapsulaba la sincronización del rendering háptico de Unity con el scheduler del dispositivo.

Esta integración habilitaba la posibilidad de sentir cualquier superficie de los objetos 3D dentro de Unity utilizando el sistema de colisiones. De las demostraciones llevadas a cabo para probar la utilidad de la integración del dispositivo háptico dentro de Unity se comprobó que se simplifica en gran medida la implementación de tareas

básicas del diseño de sistemas hápticos. Una vez trasladadas las funciones hápticas al entorno Unity las tareas del diseño del sistema se sintetizan mayormente en definir colisionadores y configurar propiedades hápticas de los objetos. Sin embargo, aún era necesario escribir código para realizar tareas más complejas.

La segunda integración del Phantom Omni en Unity la encontramos más recientemente en [26] done proveen un plugin desarrollado en C++ que permite realizar interacciones hápticas usando C# en el motor de videojuegos Unity. La interacción háptica es soportada por el toolkit OpenHaptics de Geomagic (anteriormente conocido como Sensable Technologies) y por lo tanto es compatible con toda la gama de productos que ofrecían en ese momento.

Una característica destacable de la implementación de este plugin es que abstrae las operaciones de rendering háptico estableciendo modos de interacción, propiedades hápticas de los objetos virtuales y efectos hápticos. Los modos de interacción proveen efectos de ambiente básicos y cada uno las siguientes funcionalidades: solo contacto háptico, contacto y manipulación de objetos, contacto y simulación de efectos de vibración, contacto con efectos de puntura e inyección. Por otro lado, implementa sobre los objetos virtuales las propiedades hápticas básicas definidas en OpenHaptics como son rigidez, amortiguación, fricción estática y dinámica, y agrega además propiedades inherentes a la manipulación de objetos virtuales como por ejemplo la masa del objeto, grado de manipulación, inercia de movimiento y fricciones de perforación. Por último, los efectos hápticos se aplican sobre el dispositivo háptico y sirven para transmitir efectos variados como efectos de resorte, de viscosidad, vibraciones, etc.

Las investigaciones realizadas dejan constancia de la utilidad en la estrategia de trasladar las herramientas hápticas al motor Unity. Quedó demostrado luego de realizar los prototipos que la creación de entornos hápticos inmersivos con la integración del dispositivo en Unity se reduce a tareas sencillas.

4 REQUERIMIENTOS

Como hemos visto en la sección anterior, aún queda mucho para explorar en el campo de las herramientas de autoría de háptica. Las soluciones propuestas para feedback vibrotáctil se han basado en la construcción de interfaces gráficas nuevas para alcanzar propósitos específicos. En esta herramienta nos hemos propuesto retomar los esfuerzos realizados en los editores de háptica anteriores basados en curvas, trasladando la edición a un motor de videojuegos. Además, el enfoque principal lo hemos puesto en el funcionamiento de la herramienta dentro de un entorno de desarrollo de realidad virtual.

A través de las siguientes secciones, definiremos la caracterización de los usuarios a los que va dirigida la herramienta. Luego a partir de esto describiremos las funcionalidades que deben ser provistas, y por último designaremos los elementos necesarios que componen el entorno de ejecución de la herramienta.

4.1 USUARIOS

En la caracterización de usuarios definida por Swindells *et al.* [14] se registran tres tipos potenciales usuarios para una herramienta de autoría háptica:

- Programadores: Requieren la mayor cantidad de flexibilidad y control sobre la herramienta. Tienen mayor predisposición para aprender y usar interfaces de usuario más complicadas y desarrollar código para crear efectos hápticos más sofisticados.

- Diseñadores: Tienen entendimiento técnico, pero pueden no tener habilidades en programación o predisposición a su empleo. Requieren herramientas para iterar rápidamente y realizar ajustes finos de los efectos hápticos. Usualmente prestarán más atención a conceptos de más alto nivel que los programadores.
- Usuarios finales: Incluye los usuarios de sistemas que contienen componentes hápticos como pueden ser celulares, automóviles o paneles de control. Tienen predilección por focalizarse en personalizaciones pre-definidas para su dispositivo en particular. Como por ejemplo un selector de tono vibrotáctil para llamadas en un teléfono celular.

Nuestro trabajo está centrado en los usuarios aquí caracterizados como diseñadores. La herramienta debe ser lo suficientemente flexible para especificar estímulos complejos sin necesidad de escribir código. Se debe proveer la capacidad de iterar rápidamente entre especificación de estímulos y poder probarlo en el dispositivo final.

4.2 Funcionalidades

Esta tesina está dirigida a construir una herramienta de autoría de interacción háptica para realidad virtual orientada a diseñadores. Existen diversas fuentes que describen requisitos fundamentales de editores e interacciones hápticas que sirven de soporte para nuestra selección de funcionalidades. En el trabajo de Swindells et al. [14] se recopilan principales atributos requeridos para construir herramientas de autoría háptica. Diversas prácticas de diseño de interacción háptica han sido recopiladas por MacLean K. y Hayward V. [27] y por Moussette C. [28]. Un análisis sobre principios de creación de entornos virtuales puede encontrarse en el trabajo de Rix J. et al. [29].

Para nuestro propósito, retomamos principios de prácticas de diseño háptico adaptándolos con la finalidad de extender un entorno de desarrollo de realidad virtual. Con este fin, definimos a continuación las características que debe proveer la herramienta según un esquema de prioridad.

Propiedades necesarias

Estos requerimientos son esenciales para el sistema.

- Facilitar la creación y manipulación de patrones de interacción táctil dentro de una plataforma de desarrollo de entornos virtuales.
- Proporcionar la capacidad de diseñar estímulos complejos que involucren varios actuadores en una línea de tiempo.
- La herramienta debe ser manipulable por un diseñador de interacción sin necesidad de escribir código.
- Proveer resultados que puedan ser probados con facilidad y den un reconocimiento inmediato al diseñador de lo que está generando.
- Implementar un esqueleto de una mano virtual que disponga de un mecanismo para asociar la disposición física de los actuadores vibrotáctiles del dispositivo háptico.
- Proponer mecanismos de composición de objetos del mundo virtual para dotarlos de funciones hápticas.
- Realizar pruebas de concepto donde se pueda apreciar las capacidades provistas por la herramienta.

Propiedades deseables

Estos requerimientos deberían ser satisfechos por el sistema final implementado.

- Minimizar la curva de aprendizaje requerida para generar interacciones hápticas. La herramienta debe estar embebida en una plataforma de desarrollo que resulte familiar a un desarrollador de aplicaciones de realidad virtual.
- Que el entorno de trabajo sea accesible y fácilmente replicable. Los dispositivos de hardware deben ser de acceso comercial o que se puedan construir económicamente y la plataforma de desarrollo, de manera similar, que sea gratuita o accesible.
- Proveer un mecanismo para recibir feedback visual o auditivo de los estímulos para sustituir el dispositivo háptico durante el diseño del mismo o durante la ejecución de pruebas.

PROPIEDADES CONCEBIBLES

Requerimientos que sería bueno tener, pero que no representan funciones principales del sistema final.

- Proporcionar la capacidad de diseñar estímulos que representen texturas o que puedan describir los materiales de los objetos virtuales.
- Que la disposición de los actuadores pueda ser configurada para trabajar sobre otros espacios del cuerpo.

4.3 ENTORNO DE EJECUCIÓN

4.3.1 INTRODUCCIÓN

En gran parte de los editores de feedback vibrotáctil el entorno de ejecución está restringido a la única tarea de proveer la reproducción de los estímulos. En otros casos directamente no existe entorno de ejecución, en donde solo se provee la exportación de los estímulos en

un formato legible como XML para ser utilizado en herramientas de terceros. En nuestro caso nos hemos propuesto construir el editor háptico de forma que esté completamente integrado con el entorno de ejecución de los estímulos que sean diseñados con esta herramienta. De este modo, el editor no solo provee las herramientas adecuadas para diseñar los estímulos hápticos, sino que también permite asociar los objetos virtuales que representa y las interacciones que disparan su ejecución.

En las siguientes secciones daremos un resumen del contexto actual de los dispositivos hápticos disponibles para desarrollar en realidad virtual, estableceremos los requisitos sobre cómo debe estar conformado el entorno de ejecución y, por último, describiremos cuales son las plataformas y dispositivos que conforman el entorno de ejecución.

4.3.2 Contexto actual del desarrollo en realidad virtual

En la actualidad las plataformas de realidad virtual de uso masivo se encuentran en estado de incubación, con un hito principal marcado por la irrupción en el mercado del casco de realidad virtual Oculus Rift. Los primeros prototipos de este casco aparecieron en el año 2012 bajo la denominación de *Development Kit* y tuvieron buena aceptación en la comunidad de *early adopters*. La versión final de consumidor se lanzó en abril de 2016, para ese entonces otras compañías sumaron sus propuestas como es el caso del HTC Vive y Sony Morpheus.

A la tendencia inicial marcada por los cascos de realidad virtual le siguieron el lanzamiento de dispositivos complementarios que, si bien siguen siendo de carácter experimental al día de hoy, han mostrado grandes avances y tienen buena reputación en un nicho del mercado. Entre esos dispositivos se encuentra el controlador Leap Motion que existía previamente a los cascos de realidad virtual como dispositivo

de escritorio para reconocimiento de gestos con las manos. Luego se demostró su utilidad integrado a los cascos de realidad virtual y se lanzó una adaptación llamada Orion para funcionar encastrada al casco. De este modo posibilitaron la ampliación de la sensación de inmersión al agregar las propias manos del usuario en la interacción virtual.

De las plataformas de desarrollo para realidad virtual no hay reportes disponibles sobre cómo están divididas las porciones de mercado de adopción por desarrolladores. Sin embargo, existen varios indicadores que nos permiten ilustrar la perspectiva actual. Según encuestas realizadas en 2016 por Virtual Reality Developers Conference sobre 500 profesionales involucrados en la industria de realidad virtual [30], los dispositivos sobre los que más se desarrollan experiencias de realidad virtual son: HTC Vive y Oculus Rift dominando el mercado de escritorio; y Samsung Gear VR y Google Cardboard en el mercado de dispositivos móviles. Luego tenemos que estos dispositivos proveen kits de desarrollo de manera similar a como son provistos comúnmente en la industria de videojuegos, es decir con SDKs en C++ y con integraciones en motores de videojuegos. El mercado de aplicaciones y juegos de realidad virtual se encuentra todavía en estado embrionario, con solo dos tiendas online y pocos productos disponibles. Esto hace que no se puedan obtener reportes de desarrollo concluyentes, pero si tomamos como referencia el mercado mobile y trazamos un paralelismo con el mercado de realidad virtual, podemos ver que las plataformas de desarrollo dominantes son Unity3D y Unreal Engine según reportes de Unity [31], las cuales poseen también soporte para realidad virtual.

Con respecto a los dispositivos hápticos, ha habido algunas integraciones interesantes a la experiencia visual proporcionada por los cascos. Sin embargo, los avances en este campo se encuentran atrasados. Por ejemplo, cascos como el HTC vienen con controles de captura de movimiento que proveen feedback vibrotáctil mediante actuadores montados sobre el controlador. Del mismo modo Oculus

anunció un dispositivo llamado Oculus Touch de características similares. Otros controles clásicos como los *trackpads* puedes ser utilizados con los cascos los cuales tienen cualidades hápticas similares, basadas en un conjunto de actuadores sobre el dispositivo. Estas aproximaciones las consideramos inapropiadas como métodos hápticos para el uso libre de las manos, aunque como veremos más adelante, existen alternativas experimentales que pueden ser utilizadas con esta finalidad.

4.3.3 Requisitos

En concordancia con las funcionalidades que hemos propuesto para el editor de háptica y según el estado actual de tendencias en dispositivos de realidad virtual que hemos descripto, establecemos a continuación los requisitos para el esquema de plataformas y dispositivos que utilizamos como entorno de ejecución de la herramienta de edición háptica.

En las funcionalidades necesarias de la herramienta de autoría hemos establecido orientarnos al diseño de estímulos táctiles para las manos. Para que el feedback háptico resulte inmersivo dentro de un medio de realidad virtual utilizando las manos establecemos como requisito utilizar un guante con actuadores vibrotáctiles como dispositivo de salida de recepción de feedback háptico para el usuario. Los guantes táctiles son más apropiados que los dispositivos de force feedback cuando la simulación de realidad virtual requiere destreza, libertad de movimiento, e información del estado de agarre de objetos y texturas, pero no información del peso [32]. Estos guantes son más livianos que los de force feedback y típicamente utilizan vibradores electromecánicos para transmitir información de texturas. Los actuadores son pequeños y pueden ser ubicados directamente sobre el guante. La colocación de actuadores en los lugares del guante

donde se necesita feedback táctil genera un diseño sencillo y reduce el peso y el costo del sistema.

Una vez establecido el guante como medio de feedback vibrotáctil, establecemos como requisito contar con un mecanismo complementario de tracking de movimiento o reconocimiento de gestos como método de entrada para conocer la posición y gestos de la mano. De este modo, los estímulos hápticos pueden ser disparados en función de la acción del usuario con sus propias manos en libertad de movimiento, logrando un grado de inmersión ideal en realidad virtual.

Por último, pero no menos importante, establecemos como requisito el uso de un casco para completar el esquema de dispositivos en un entorno de realidad virtual. De esta manera, establecemos un entorno de ejecución apto para realidad virtual dotado de un medio visual, uno háptico y de detección de gestos. El medio auditivo si bien no lo establecemos como requisito porque excede el alcance de este proyecto, lo consideramos fácil de acoplar a este entorno ya que forma parte de las funcionalidades básicas que provee un entorno de desarrollo que si establecemos como requisito como veremos a continuación.

En cuanto a la plataforma de desarrollo, la herramienta debe estar embebida en un entorno de desarrollo de realidad virtual. Tal que resulte familiar al grupo de desarrolladores que trabaja sobre ese paradigma actualmente, de modo de hacerlo accesible en el mayor espectro posible de usuarios. El entorno debe tener soporte para dispositivos de realidad virtual, contar con la posibilidad de realizar *plugins* para conectar nuevos dispositivos y tener un conjunto sólido de herramientas de extensión del editor.

Con estos requisitos en mente, en las siguientes secciones describiremos las tecnologías que han sido empleadas para configurar el entorno de ejecución de la herramienta.

4.3.4 Dispositivos

4.3.4.1 Casco de realidad virtual

Para este esquema tecnológico empleamos el casco de realidad virtual de Oculus Rift en su versión Development Kit 2. Como veremos más adelante, el soporte nativo de Unity para realidad virtual permite intercambiar el casco entre diferentes productos disponibles conservando la compatibilidad a futuro. Por lo tanto, si bien el Oculus Rift es nuestro dispositivo de referencia este puede ser reemplazado por otros disponibles actualmente en el mercado como el HTC Vive conservando intacto el esquema propuesta. En los aspectos técnicos, el Rift en su versión DK2 está compuesto por una pantalla OLED de resolución 960x1080 por cada ojo y una taza de refresco de 75Hz.

El Oculus Rift es un dispositivo de entrada y salida. Cuando el usuario mueve la cabeza, la información de movimiento, la posición de la cabeza y su inclinación son enviados a la computadora. El *runtime* de Oculus debe procesar los datos, generar la imagen con una aplicación que calcule como se representa el próximo cuadro y enviarla al dispositivo. El margen de latencia para no producir trastorno de movimiento es de 20 milisegundos.

4.3.4.2 Tecnología de reconocimiento de gestos

El dispositivo que seleccionamos para realizar el reconocimiento de gestos es el controlador Leap Motion. El cual es un sensor de gestos robusto, de bajo costo disponible en el mercado (u$d 80), con una amplia integración en tecnologías de realidad virtual y sus plataformas de desarrollo.

El sensor de Leap Motion reconoce y rastrea el movimiento de las manos y sus dedos reportando la posición, velocidad y orientación en baja latencia y alta precisión. A efectos de ser utilizado en realidad

virtual el controlador es montado en el casco utilizando una montura.

El dispositivo de Leap Motion inicialmente surgió para ser utilizado como dispositivo de escritorio y con el advenimiento de la realidad virtual dispuso una montura junto a la venta del producto y proveyó de un conjunto de guías de usabilidad y herramientas a la comunidad de desarrolladores[7]. El controlador de hardware se mantuvo igual y el salto cuantitativo se dio cuando actualizaron el software de rastreo a la versión conocida como Orion (agosto de 2014). Que constituyó una mejora notable ya que el modelo basado en escritorio está optimizado para rastrear las manos de su parte palmar al dorsal. A diferencia del modo del modo de uso basado en escritorio, el dispositivo al ser empleado en la parte frontal del casco, implica que la detección de la mano se realiza de la parte dorsal al palmar la mayor parte del tiempo. Y como el sensor se mueve con la cabeza, la posición de las manos depende del punto de vista. Con la utilización del primer SDK producía muchos errores de precisión y oclusión de los dedos de las manos que sin un modelo predictivo quedaban fácilmente desarticulados al ser tapados por las manos.

Al emplear el dispositivo montado en el casco una de las principales características que emergieron fueron la utilización de la visualización en infrarrojo de los sensores de las cámaras para combinar la experiencia de realidad virtual con una visión estereoscópica del mundo alrededor, lo que se conoce comúnmente como realidad aumentada.

El controlador utiliza sensores ópticos y luz infrarroja y tienen un campo de visión de 150 grados. Siendo que el Oculus Rift tiene un ángulo de visión de 110 grados, esto permite que el rastreo de las manos

[7] Un prototipo de sensor fue anunciado con el código “Dragonfly”. Diseñado para funcionar embebido en cascos de realidad virtual, el dispositivo muestra mejorías con un campo de visión mayor y en la resolución de la imagen. Sumado a la incorporación de cámaras con detección de color lo cual designa un adelanto en la experiencia combinada de realidad virtual con realidad aumentada.

comience antes de que el rango de visión del usuario alcance la visión de sus manos. El campo efectivo de detección se extiende de 30 a 60 centímetros por delante del dispositivo.

La comunicación con la computadora y la transformación de datos se produce entre el dispositivo de hardware y un componente de software que se ejecuta como un servicio o daemon en el sistema operativo. El componente de software procesa las imágenes capturadas por el hardware y envía la información de seguimiento a las aplicaciones. El plugin de Unity de Leap Motion se conecta a este servicio para obtener los datos. Los scripts incluidos en el plugin trasladan las coordenadas de Leap Motion al sistema de coordenadas que utiliza Unity y escalan las unidades de medidas de milímetros a metros.

4.3.4.3 Controlador de feedback vibrotáctil

Antecedentes

En trabajos anteriores sobre editores de háptica hubo diferentes aproximaciones sobre que dispositivos hápticos eran soportados, ya que no existen aún productos tecnológicos que estén establecidos y, en muchos casos, se diseñan desde cero con un propósito en particular. Como hemos visto en la sección de *Trabajos relacionados*, podemos separar los dispositivos hápticos entre los de force feedback y los vibrotáctiles como metodologías disjuntas sobre cómo realizar la reproducción de estímulos hápticos. Aun así, variaciones entre estas dos categorías de dispositivos pueden llevar a construcciones diferentes de editores.

Podemos discernir tres aproximaciones en la elección de dispositivos soportados en una herramienta de diseño háptico. Una en la cual se diseña un dispositivo de hardware desde cero para satisfacer los requerimientos del editor. Otra aproximación es utilizar dispositivos disponibles comercialmente y adaptar las funcionalidades del editor

para dichos dispositivos. Por último, es posible fijar el objetivo de construir una herramienta de propósito general que permita cubrir la mayor cantidad de dispositivos hápticos posibles.

Guante háptico

Para este trabajo haremos uso de un solo dispositivo háptico y que cumpla con los requisitos que hemos establecido. Las características necesarias que este dispositivo debe cumplir son las de proporcionar un guante con actuadores vibrotáctiles de baja latencia, de bajo costo y que sea fácilmente replicable.

Por el lado de los guantes hápticos comerciales, Burdea y Coiffet hicieron un relevamiento en 2003 de los dispositivos vibrotáctiles que había disponibles [33]. De los cuales en esa fecha solo existían los guantes que ofrece Cyberglove Systems LLC.. Estos guantes, que aún hoy se comercializan, están orientados a aplicaciones industriales por lo cual resultan muy costosos y además proveen un número reducido de actuadores lo que los hace poco útiles en trabajos de investigación. En la actualidad el panorama no es muy distinto, unas pocas soluciones han aparecido como las de ManusVR y Gloveone ofreciendo guantes vibrotáctiles para el público hogareño propiciadas por la irrupción en los últimos años de los cascos de realidad virtual. Sin embargo, estos guantes todavía no salieron a la venta, sino que solo se han dado a conocer prototipos en exposiciones.

La solución en este caso viene provista por los *toolkits* vibrotáctiles que vienen provistos en forma de *do-it-yourself* frutos de investigaciones académicas y aportes de aficionados. En particular, el *toolkit* que utilizamos como dispositivo háptico para nuestro trabajo consiste en una placa Arduino con motores vibrotáctiles integrados que recibe comandos a través de un puerto serie.

Arduino es una compañía que provee hardware libre enfocado en facilitar la electrónica y programación de sistemas embebidos en proyectos interactivos. La solución viene compuesta por circuitos impresos que integran un microcontrolador y un entorno de desarrollo. Toda la plataforma es provista con licencia de código abierto, mediante el cual los circuitos pueden ser replicados y esto propició una nutrida comunidad de desarrolladores que aporta soluciones.

En nuestro esquema empleamos una placa Arduino destinando los contactos PWM para los motores vibrotáctiles. La salida de *Pulse Width Modulation* o PWM por sus siglas, es utilizada para simular una salida analógica con rango de voltajes utilizando un medio digital. El rango de voltajes se utiliza para ampliar las posibilidades de los motores, de prendido a apagado, a simular un espectro de fuerzas táctiles que puede ser utilizado para representar curvas de fuerzas o modificar la sensibilidad del tacto percibido por el usuario. Según la cantidad de salidas de la placa Arduino, diferente cantidad de motores pueden ser conectados. En nuestra implementación la cantidad de actuadores es variable y se configura en el editor como así también la disposición física de cada uno. La comunicación entre Unity y la placa Arduino es implementada en un *plugin*. Los motores vibrotáctiles son dispuestos físicamente en un guante para actuar en distintas partes de la palma de la mano. La cantidad de motores empleados es variable y puede ser ampliada para cubrir tantas partes de la mano cómo se considere necesario.

El guante háptico resultante funciona en estrecha relación con la captación de gestos provista por Leap Motion. El controlador Leap Motion actúa como principal input de eventos del usuario, donde la posición de la mano es procesada en tiempo real por el servicio. Cuando se disparan eventos de simulación háptica se procesa el renderizado háptico y se envía la señal al guante a través del controlador el cual constituye el output resultante. El renderizado háptico se produce en la aplicación como producto del diseño de las interacciones hápticas provisto por la herramienta.

La comunicación se realiza entre Unity operando desde un dispositivo host y el microcontrolador Arduino operando como medio receptor de instrucciones. El protocolo de comunicación empleado es Firmata basado en el formato de mensajería MIDI en donde se utilizan mensajes de 2 bytes de longitud que contienen comandos y datos.

Actuadores

El factor más importante en cuanto a la variedad de sensaciones que pueden ser generadas por el guante háptico está determinado por la tecnología empleada. Los requisitos que deben cumplir los actuadores que usamos para nuestro *toolkit* son: que sean ligeros y lo suficientemente pequeños para poder adaptarse a un guante, de bajo coste, que puedan conseguirse fácilmente, y que provean un grado de sensaciones aceptable. Para este último punto las dos consideraciones que tendremos en cuenta son, por un lado, que permitan aplicar distintos voltajes para variar la fuerza de los estímulos y ajustar la sensibilidad, y en segunda instancia, que tengan un tiempo de latencia pequeño.

En aplicaciones hápticas, la latencia viene dada en que existe un retraso entre la detección del movimiento del usuario y la producción del estímulo háptico. Si este retraso es lo suficientemente alto el usuario experimenta cambios en la sensación percibida, y si ese tiempo supera cierto umbral, los usuarios terminan desasociando la sensación como causa de sus propias acciones. Estos retrasos fueron estudiados por Shogo Okamoto et al. [34] y los umbrales aceptables fueron estimados en tiempos de aproximadamente 40 y 60 milisegundos.

De las pruebas de evaluación llevadas a cabo por Muñoz [11] se desprende como resumen el cuadro que mostramos a continuación, el cual compara los aspectos a tener en cuenta para la construcción del controlador Vitaki, los cuales coinciden con nuestros requisitos. A

partir de esto establecemos el requisito de usar actuadores del tipo LRA o ERM para componer el guante háptico.

	LRA	ERM	PIEZO	SMA	BOBINAS
Tamaño	Peq.	Peq.	Variab.	Variab.	Grande
Espesor	Peq.	Peq.	Mínimo	Variab.	Grande
Latencia (ms)	20-30	40-80	<1	Alta	<1
Frecuen. (Hz)	~175	50-250	1-300	~10	200-300
Voltaje (V)	<5	<5	50-200	<5	<3
Precio (€)	5-10	1-5	50-170	Variab.	~200

Figura 6: Comparativa de distintas tecnologías comerciales de actuadores táctiles. Fuente: [10]

Los actuadores lineales resonantes (LRA) están formados por una bobina fija y un imán fijado a una masa. El imán y la masa, cuya sujeción depende de un muelle, son atraídos y repelidos por la bobina produciendo un movimiento de oscilación. Tienen un precio competitivo por unidad y un tamaño reducido [11].

Los actuadores ERM están basados en miniaturas de motores de corriente continua con una masa unida al eje. La rotación de la masa produce el desplazamiento del motor lo que genera una vibración perpendicular a su eje de rotación.

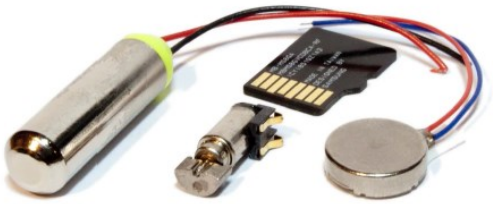


Figura 7: Diferentes actuadores ERM disponibles en el mercado.

Actualmente pueden encontrarse muchos usos comerciales en dispositivos de feedback vibrotáctiles basados en ERM. Varios *gamepads*,

incluyendo los de las consolas de videojuegos más populares, incorporan feedback vibrotáctil con tecnología ERM. La librería Haptic SDK de Immersion provee decenas de efectos para actuadores ERM para ser usados en juegos de dispositivos móviles. También se pueden encontrar guantes de realidad virtual con actuadores vibrotáctiles ERM como el dispositivo CyberTouch de la empresa CyberGlove Systems.

Uno de los puntos desfavorables de usar actuadores ERM es su relativamente alta latencia que son de aproximadamente de 200 ms para arrancar y 250 ms para frenar completamente. Con las técnicas de *overdrive* y frenado estos retrasos se reducen a menos de 50 ms.

La técnica de *overdrive* acelera el tiempo de arranca del actuador ERM. El tiempo de arranque es proporcional al voltaje aplicado. Si bien cada actuador tiene valores de voltajes operativos determinados de fábrica, si se aplican pulsos cortos de mayor voltaje al permitido se produce una mayor aceleración sin riesgo de sobrecalentamiento. Estos pulsos cortos de mayor voltaje son los llamados *overdrive,* que acortan el tiempo de arranque y por lo tanto reducen la latencia.

La técnica de frenado activo mejora el tiempo de desaceleración y contribuye a dejar los tiempos de latencia dentro del rango aceptable. La técnica consiste en la aplicación de corriente al motor con la polaridad revertida. Esto produce un freno efectivo para la masa excéntrica del motor reduciendo el tiempo de parada significativamente.

Los valores especificados de vibración son genéricos, de modo que no dependan del motor de vibración usado en el momento. Por lo tanto, los valores de vibración no son expresados en voltaje. En cambio, una unida flotante normalizada es empleada con un rango de entre -1.5 a 1.5. Los valores expresados entre 0 y 1 corresponden al voltaje nominal mínimo y máximo de los actuadores, mientras que los valores 1.0 a 1.5 son escalados a los voltajes de *overdrive*. Los valores negativos

tienen la misma consideración, solo que son usados para propósito del frenado activo.

4.3.5 Plataformas de desarrollo de realidad virtual

Como plataforma de desarrollo para establecer la herramienta de háptica haremos uso de un motor de videojuegos. Estos consisten típicamente en un *framework* con funcionalidades integradas propias del desarrollo de videojuegos, como suelen ser un motor de *rendering* para gráficos 3D o 2D, un motor de física o detección de colisiones, un gráfico de escena, funciones de animación y cinemática, entre otras cosas. Para nuestros requisitos el motor tiene que tener soporte para dispositivos de realidad virtual y que cuente con la capacidad de poder extender el editor.

4.3.5.1 Unity3D

Unity es un motor de videojuegos multiplataforma propietario que ofrece una licencia gratuita con acceso a gran parte de sus prestaciones. En los últimos años se ha establecido como el motor más utilizado por los desarrolladores, con 1 millón de usuarios mensuales [35] lo que lo convierte en la mayor comunidad de desarrollo tanto para móviles como en realidad virtual.

Soporte de realidad virtual

El soporte para realidad virtual estuvo en principio ligado a *plugins* desarrollados por terceros para dar soporte a los dispositivos. A partir de la versión 5.1 [36] provee soporte integrado para gran variedad de dispositivos de realidad virtual. Esto significa un marco de desarrollo simplificado al evitar administrar los *plugins* de terceros,

dando soporte hacia atrás con nuevos lanzamientos de dispositivos y simplificando la configuración de transiciones entre el modo de uso para realidad virtual y pantalla estándar [37].

Cuando se activa el soporte de Unity VR todas las cámaras[8] de la escena renderizan directamente al casco. La vista y la matriz de proyección es automáticamente ajustada para el seguimiento de la cabeza, la posición del cuerpo humano y el campo de visión correspondientes al casco de realidad virtual que corresponda. Para dar soporte a la visión estereoscópica se utilizan dos cámaras asignando una por cada ojo, lo que representa un *rendering pipeline* de dos *frames*. En las cámaras se definen la distancia de separación de los ojos y la distancia de convergencia.

EXTENSIBILIDAD DEL EDITOR

Unity tiene la cualidad de proveer variables públicas para configurar los objetos virtuales de la escena a través de un inspector de objetos. Este recurso es típicamente utilizado para controlar el diseño del comportamiento de los objetos en escena, pero también se puede utilizar para crear herramientas. Además de estas características, Unity dispone de capacidades para extender el editor a través de una API diseñada para este propósito, el cual provee ventanas y controles de GUI personalizables.

La integración de *plugins* para extender las funcionalidades por fuera del editor está provista por dos tipos de librerías: los *managed plugins* desarrollados con .NET los cuales comparten acceso a las herramientas provistas por Unity que también utiliza el mismo *frame-*

[8] La cámara en Unity es un componente que captura y visualiza el mundo al usuario. En ella se definen como es la simulación de perspectiva, el campo de visión y los planos de corte, entre otras cosas.

work; y los *native plugins*, que son librerías de código nativo específicas para cada plataforma que pueden estar escritas en lenguajes como C, C++, Objetive C, etc.

4.3.5.2 Unreal Engine

Otro motor de videojuegos con soporte para realidad virtual es Unreal Engine. De características similares a Unity, este motor goza de mayor popularidad en un sector más pequeño del mercado que es el de los videojuegos de alto presupuesto para consolas hogareñas. Para facilitar el desarrollo, Unreal cuenta con un sistema de comandos visual llamado *blueprints* el cual está basado en el concepto de interfaces de nodos para conectar componentes del juego, lo que permite crear muchos aspectos del juego sin necesidad de escribir código. Aun así, la arquitectura y la API del motor están basadas en C++ lo cual significa un incremento en el nivel de complejidad para construir herramientas por tener que lidiar con aspectos de bajo nivel como administración de memoria.

Soporte de realidad virtual

Unreal tiene un soporte amplio para dispositivos de realidad virtual. A partir de la versión 4.1.1 lanzada en abril de 2016 provee soporte nativo para Oculus Rift y Leap Motion, siendo junto con Unity los únicos en la actualidad con soporte integrado para estas plataformas.

Extensibilidad del editor

La extensión del motor está basada en un *framework* de programación de personalización de interfaces gráficas llamado StateUI. El *framework* está diseñado para construir interfaces de usuario para herramientas del editor usando una sintaxis declarativa. Más aun, es

posible acceder al código fuente del motor, con lo cual las posibilidades de extensión son ilimitadas.

4.3.5.3 *Conclusiones*

Los motores descriptos anteriormente son los que gozan de mayor aceptación en la comunidad de desarrolladores de realidad virtual. Además, en nuestro caso, solo estos son los que tienen soporte para los dispositivos que seleccionamos para desarrollar la herramienta.

Para este trabajo hemos elegido Unity como plataforma de desarrollo para implantar nuestra herramienta por varias razones. Lo que lo constituye de nuestro mayor interés, es la disponibilidad de recursos y la base de conocimientos que posee para desarrollos en realidad virtual. Asimismo, tanto Oculus Rift como Leap Motion tienen la mayor parte de sus recursos para desarrolladores disponibles para Unity por sobre otros motores de videojuegos.

4.4 Especificación de requisitos

Según la descripción de funcionalidades y el entorno seleccionado a continuación definimos la lista de tareas correspondientes al proyecto de la herramienta de autoría. Utilizamos un esquema de *backlog* de metodologías ágiles para delinear una serie de historias de usuarios que deben ser implementadas para construir la herramienta. Estas historias representan requisitos en lenguaje común de usuario las cuales luego son divididas en tareas específicas para el desarrollo cotidiano.

Este método resulta apropiado para planificar el proyecto de desarrollo dado que está comprobado que las metodologías ágiles resul-

tan idóneas cuando las condiciones iniciales presentan muchas incertidumbres [38]. En nuestro caso las tecnologías del ecosistema de la herramienta son de carácter experimental con lo cual el software *third-party* provisto se actualiza muy seguido y fluctúa bastante en funcionalidades provistas y estabilidad. Además, la herramienta que construimos es de carácter experimental y como no está basado en otro software ya existente resulta deseable tener funcionalidades emergentes.

En el esquema empleado se realizaron iteraciones en donde se implementaron las historias según el orden de prioridad en ciclos completos de diseño, implementación y análisis.

ID	Tema	Como…	Quiero…	Tal que…	Notas	Prioridad
1	Editor	Diseñador	configurar la disposición de actuadores en la mano	se correspondan los actuadores del editor con los del guante físico		Alta
3	Editor	Diseñador	diseñar un patrón háptico por fuerza aplicada en el tiempo	pueda crear estímulos que describan contacto simple	Se utilizan curvas para representar el estímulo	Alta
4	Editor	Diseñador	diseñar un patrón háptico por profundidad de penetración	pueda crear estímulos basados en la fuerza a aplicar según el grado de penetración	Se utilizan curvas para representar el estímulo	Media

2	Editor	Diseñador	diseñar un patrón háptico utilizando varios actuadores al mismo tiempo	pueda crear un estímulo que emita múltiples fuerzas en simultáneo	Se utiliza una línea de tiempo para diseñar múltiples curvas	Media
8	Editor	Diseñador	reutilizar las curvas que definen los estímulos hápticos	poder replicar efectos rápidamente		Media
10	Editor	Diseñador	ver las fuerzas hápticas aplicadas gráficamente en las manos virtuales en modo edición	tener feedback instantáneo mientras se diseña el estímulo	Se utilizan gamas de colores según la intensidad de la fuerza	Baja
5	Editor	Diseñador	dotar de propiedades hápticas objetos de mi escena	pueda corresponder los estímulos hápticos con los objetos e interacciones que los desencadenan		Alta
6	Editor	Diseñador	probar la escena en play mode y sentir los estímulos hápticos	tener feedback preciso del diseño realizado en el editor	Las pruebas son con el guante háptico y el Leap Motion funcionando	Media

7	Editor	Diseñador	probar la escena en play mode y ver los estímulos reflejados gráficamente en la mano virtual	tener feedback rápido sin colocar el guante	Solo se necesita el Leap Motion funcionando	Baja
9	Editor	Diseñador	que se guarden todos los cambios realizados en el proyecto	poder persistir el trabajo realizado		Media

5 DISEÑO CONCEPTUAL

A partir de los requisitos establecidos en la sección anterior describiremos a continuación los elementos del diseño conceptual de la herramienta háptica. Estos elementos sirven como esquema de alto nivel del funcionamiento de la herramienta, definen la arquitectura del sistema y se introducen conceptos abstractos de la forma de operar. En el capítulo que sigue a este, se describe la implementación de la herramienta y como estos conceptos son puestos en práctica.

5.1 ARQUITECTURA DEL SISTEMA

El sistema que vamos a construir consiste fundamentalmente en cinco componentes interrelacionados. La siguiente figura muestra la relación entre estos componentes y a continuación describimos la función que tiene asignado cada uno en la arquitectura de nuestro sistema.

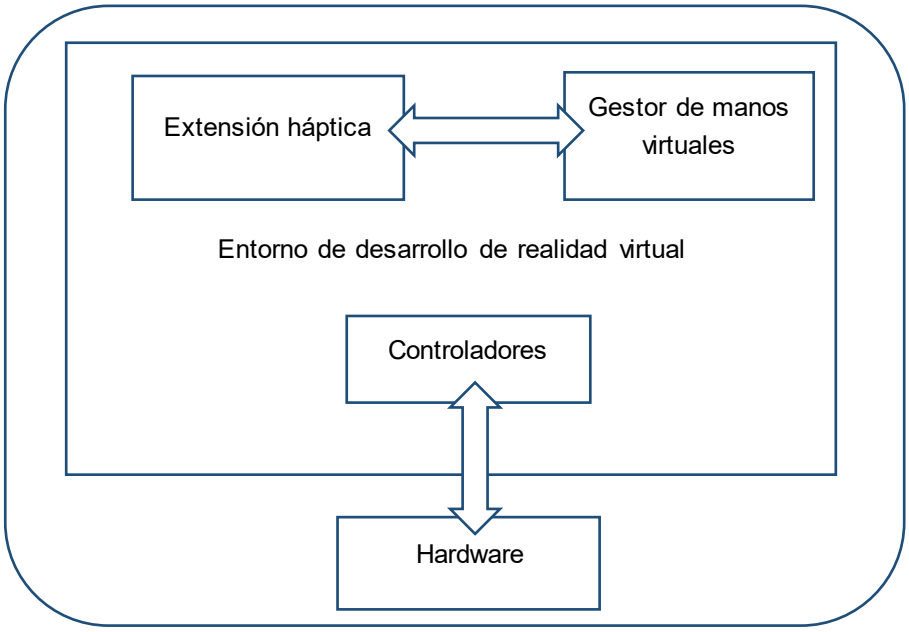


Figura 8: Componentes del sistema.

5.1.1 Entorno de desarrollo de realidad virtual

En este componente designamos el software de desarrollo que implementa las funcionalidades básicas para crear entornos virtuales, que en nuestro caso es el motor de videojuegos Unity que hemos seleccionado. El motor funciona como un ecosistema ya conocido para los desarrolladores de entornos virtuales, el cual extendemos para dar soporte al diseño de patrones hápticos en conjunción con los dispositivos que componen la experiencia virtual.

5.1.2 Gestor de manos virtuales

Establecemos dos capas en el uso de manos virtuales. En este componente designamos la capa de más bajo nivel que es la de creación y administración de manos virtuales, el segundo componente es la extensión háptica. La extensión háptica opera en conjunción con los

componentes provistos por Leap Motion, los cuales tienen un flujo de trabajo establecido para desarrollar manos virtuales. Reutilizar el flujo de trabajo de estos componentes resulta idóneo para nuestro proyecto dado el progreso que ha tenido durante múltiples iteraciones en las cuales han ido mejorando el proceso de diseñar manos en entornos virtuales.

Los componentes provistos por Leap Motion, llamados *Unity Core Assets,* están organizadas en un conjunto de recursos modulares sobre los cuales se pueden ir agregando accesorios para expandir las funcionalidades básicas. Los componentes del núcleo proveen un flujo de trabajo para diseñar manos en entornos virtuales y una API para comunicar con el dispositivo de hardware. A continuación, mostramos un esquema de los componentes para crear manos en Leap Motion.

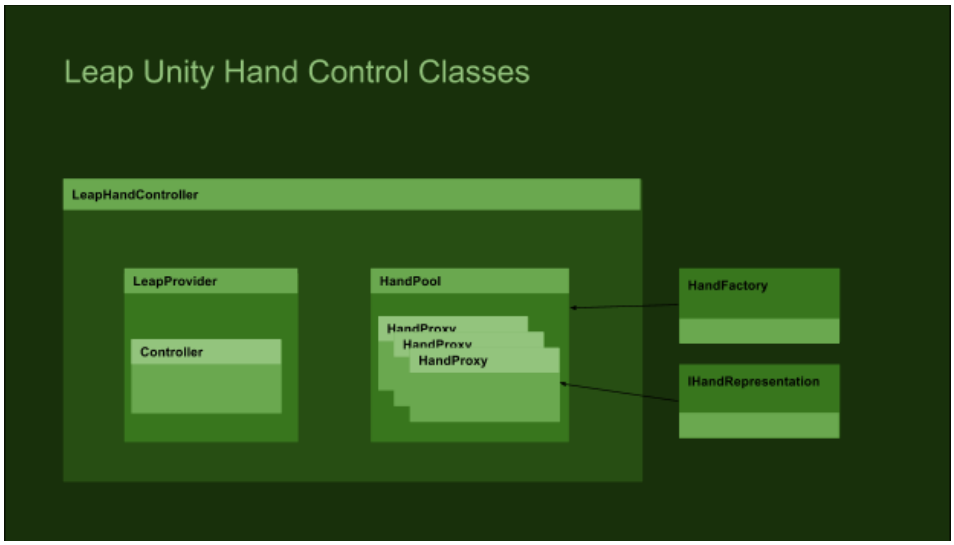


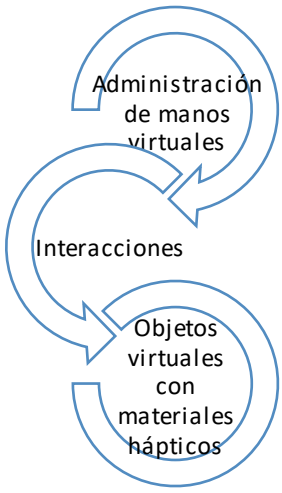
Figura 9: Estructura de clases en Leap Motion Orion. Fuente[9]

[9] http://blog.leapmotion.com/redesigning-our-unity-core-assets-new-workflow-and-architecture-for-orion/

El esquema provisto por Leap Motion está basado en el patrón *factory* en el cual la clase *LeapHandController* actúa como ensamblador, la clase *HandPool* actúa como factoría y produce objetos del tipo *HandRepresentation*. Estos últimos combinan un modelo de manos 3D acoplados con scripts que permiten conducirlos utilizando los datos del dispositivo.

5.1.3 Extensión háptica

Definimos extensión háptica como los componentes nuevos que introducimos para dotar al esquema con las propiedades que establecimos para diseñar patrones e interacciones hápticas. Este componente consta de dos partes: por un lado, se extienden las representaciones de manos de Leap Motion para crearlas con propiedades hápticas y, separado de este, se establecen los mecanismos para asociar materiales hápticos a los objetos virtuales y las interacciones que pueden ocurrir con las manos. Al separar la creación y administración de manos del resto buscamos, como objetivo secundario, facilitar el uso de los patrones hápticos en otras partes del cuerpo.



5.1.4 Controladores

En este componente designamos las librerías que conforman el ensamble entre el entorno de desarrollo y el hardware subyacente. Los

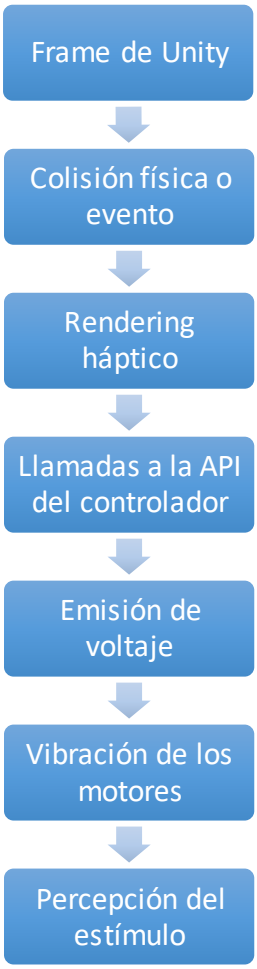
dispositivos que no son soportados de manera nativa por el editor serán agregados mediante plugins de Unity y utilizando interfaces con puerto series.

5.1.5 HARDWARE

El hardware subyacente son todos los dispositivos de realidad virtual que, conectados a la computadora, sirven para reproducir y testear la herramienta. Estos dispositivos dialogan con la computadora a través de *runtimes* o *drivers,* y luego con el motor de videojuegos mediante *plugins*. En el caso de los cascos de realidad virtual como es el Oculus Rift, son soportados de forma nativa por la plataforma sin necesidad de *plugins*. Siguiendo el modo de trabajo en Unity, el entorno virtual incluyendo los estímulos hápticos son testeados utilizando el modo de juego (Play Mode).

Según el esquema propuesto, los periféricos que dialogan con la computadora efectuando la comunicación hombre/maquina se conforman como describimos a continuación. El casco de realidad virtual Oculus Rift opera como dispositivo de entrada proveyendo la posición y orientación de la cabeza del usuario. Al mismo tiempo funciona como dispositivo de salida dando el feedback visual al usuario mediante la pantalla incorporada y un juego de ópticas que dan un rango de visión de 110 grados. Luego, el detector de gestos Leap Motion va montado sobre la parte frontal del casco de realidad virtual y funciona como dispositivo de entrada mediante el seguimiento de la posición y orientación de la mano, el antebrazo y los dedos. Por último, el controlador Arduino, con la disposición de actuadores del guante, actúa como dispositivo de salida produciendo el feedback háptico sobre las manos del usuario.

5.2 PIPELINE DE FEEDBACK HÁPTICO



Con pipeline de feedback háptico referimos a la secuencia de pasos que ocurren para crear un estímulo táctil proveniente del entorno virtual. En esta sección describimos cual es el esquema que utilizamos para reproducir el feedback háptico, ya que la herramienta propuesta contempla la etapa de reproducción de los estímulos diseñados. En el siguiente diagrama mostramos una síntesis del pipeline háptico y a continuación haremos una descripción de cada uno de los pasos.

El primer paso del pipeline es la ocurrencia de un evento en el mundo virtual que desencadena el estímulo háptico. Los eventos que hemos predefino para desencadenar los patrones hápticos son por contacto. Dentro de Unity los contactos suceden cuando el motor de física detecta colisiones dentro de cada *fixed frame* del *loop* principal de Unity. El motor de física provee los componentes que llevan adelante la simulación física, posibilitando de este modo el comportamiento pasivo de objetos de un modo realista. De esta manera, los eventos que pueden desencadenar estímulos son cuando la mano virtual toca objetos o cuando la mano ingresa en un volumen previamente definido. Estos eventos predefinidos dejan abierta la posibilidad creativa de configurar estímulos hápticos simulando propiedades táctiles de los objetos como texturas, rigidez, o incluso utilizar áreas enteras donde no

hay contacto en puntos específicos para simular sensaciones como puede ser el contacto con agua o fuego.

Cuando se desencadena un estímulo se produce el *rendering* háptico que consiste en trasladar la representación visual del estímulo diseñado en la herramienta de autoría, que en nuestro caso son las curvas, a una secuencia de llamadas a la API del dispositivo con el rango de valores aceptados para representar las fuerzas producidas en cada punto del tiempo y en que motor acontecen. La API traduce las instrucciones de fuerzas en valores de voltaje admitidos por los motores vibrotáctiles empleados. Por último se produce en consecuencia la vibración de los motores y el usuario percibe la sensación táctil.

5.3 Proceso de diseño háptico

El uso de la herramienta está pensado para funcionar con un flujo de trabajo que comienza con la configuración de los actuadores y luego da comienzo a ciclos de diseño y prueba de patrones hápticos. El ciclo iterativo es necesario porque la representación de las señales en curvas no es directamente trasladable a la sensación que se siente en la piel cuando el efecto se dispara en los actuadores. Por eso es necesario iterar continuamente entre el ajuste detallado de las curvas y la prueba del estímulo resultante. Aun así, como hemos visto, las curvas son un mecanismo conveniente y flexible para diseñar los estímulos. Una vez identificados los principales procesos, será fundamental acotar el tiempo total que conlleva cada uno para que la herramienta resulte efectiva.

En el siguiente diagrama de casos de usos establecemos las actividades que el diseñador de entornos virtuales realiza con la herramienta de autoría.

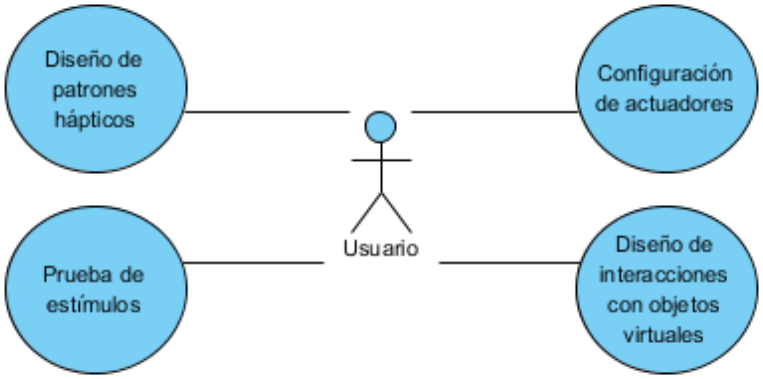


Figura 10: Actividades de trabajo en la herramienta

CONFIGURACIÓN DE ACTUADORES

La configuración de actuadores refiere al proceso según el cual el diseñador establece en la herramienta la disposición de actuadores en la mano virtual en simetría con la disposición física de los motores vibrotáctiles en el guante. La disposición de los actuadores, en nuestro caso, lo hemos restringido a las partes de la mano para que trabaje conjuntamente con el tracking de movimiento de manos en realidad virtual. Por eso establecemos que el proceso de configuración de actuadores funcione acoplado al esquema de representación de manos virtuales de Leap Motion tal como hemos detallado previamente. En este caso, la representación física de las manos virtuales que reutilizamos está compuesta por los huesos de las manos y los actuadores son asociados a cada uno de esos componentes.

Diseño de interacciones con objetos virtuales

El diseño de interacciones háptico consiste fundamentalmente en dotar objetos virtuales con propiedades hápticas y en la construcción de una especificación de las mismas. Así como se incorpora sonido a una escena virtual, acoplando componentes de sonido a determinados objetos, del mismo modo se incorpora háptica utilizando el motor de videojuegos. Por la forma en que funciona la composición de objetos, la incorporación de háptica puede ser creando un componente desde cero o reutilizando uno existe, ya sea desde la copia de un objeto en escena o acoplándolo desde una colección de componentes hápticos. Las interacciones que se pueden establecer son entre las manos del usuario cuando entran en contacto con un objeto virtual con propiedades hápticas, éstas pueden ser simples estímulos locales al tocar objetos o desencadenamientos de estímulos complejos como parte de efectos preestablecidos.

Diseño de patrones hápticos

El proceso de diseño háptico consiste en la creación de estímulos hápticos usando curvas y en la configuración de las interacciones que disparan dichos efectos. Este proceso es donde se incorpora el diseño de interacciones hápticas y, por lo tanto, se encuentra desacoplado de la representación de las manos, funcionando como un componente nuevo al esquema de desarrollo de entornos virtuales establecido por el uso de cascos virtuales y tracking de movimiento.

Los elementos y mecanismos utilizados en el diseño de estímulos hápticos mantiene estrecha relación con aquellos empleados en la creación de videojuegos y de entornos de realidad virtual en general. Por esa razón, hemos incorporado el editor de curvas que provee Unity para especificar los estímulos y utilizamos composición de objetos en la escena para configurar los materiales hápticos y las interacciones que los desencadenan. De este modo, buscamos que el diseñador

pueda crear estímulos hápticos con elementos que le resulten familiares. Esto reduce la curva de aprendizaje de la herramienta de autoría háptica ya que los mecanismos resultan más intuitivos al usuario.

Prueba de estímulos hápticos

El proceso de probar los estímulos, en similitud con el resto de los procesos, sigue los lineamientos de Unity para el desarrollo de entornos de realidad virtual. Para esto, el usuario de la herramienta prueba los resultados en el mismo editor utilizando el *play mode* de Unity. Al activar el *play mode* los dispositivos de hardware se activan, incluyendo el controlador de feedback háptico que opera a través del *plugin*, y el diseñador es capaz de probar al instante los resultados sin necesidad de compilar el código ni de depender de la creación de un ejecutable.

De modo de minimizar la dependencia del diseñador con el guante, se utiliza una mano virtual que reacciona a los estímulos hápticos de manera visual utilizando colores sobre el modelo gráfico de la mano. El uso del modelo gráfico de la mano representando estímulos es mucho más veloz en las pruebas ya que evita la colocación del guante. Sin embargo, es solo una referencia ya que no dispara estímulos hápticos realmente y por lo tanto las pruebas definitivas deben hacerse con el guante puesto.

6 IMPLEMENTACIÓN

En esta sección describimos los detalles de implementación de la herramienta de autoría háptica presentada en este trabajo la cual fue realizada en base a los requerimientos previamente propuestos.

6.1 INTRODUCCIÓN

La herramienta de autoría funciona como una extensión del motor de videojuegos Unity y además trabaja en conjunción con las herramientas de desarrollo de realidad virtual provistas por Oculus y Leap Motion. Por lo tanto, para entender el funcionamiento de la herramienta describiremos el contexto del ecosistema donde se encuentra alojado.

Unity es un entorno de desarrollo con muchas funciones incorporadas orientadas al prototipado rápido de videojuegos como son un motor de *rendering* gráfico, un motor de física, un gráfico de escena, entre otros. Los elementos de abstracción básicos en que se funda Unity para la creación de entornos virtuales pueden describirse a partir de los conceptos de GameObject y de Component. Los GameObjects son, en principio, todos los elementos que pertenecen a la escena. La particularidad de estos elementos es que la forma de extender su comportamiento es mediante componentes (Component) que proveen módulos de código y pueden ser reutilizados en otros objetos. Estos componentes permiten dotar de comportamiento rápidamente a los objetos de escena y configurarlos de forma sencilla usando *widgets* visuales. La implementación real de estos componentes está basada en *scripting*.

En este sentido, Unity puede ser considerado un sistema de software basado en componentes. Sin embargo, es importante aclarar que los lenguajes de programación que soporta de manera nativa son lenguajes orientados a objetos. Esto se explica mediante las abstracciones del motor explicadas anteriormente que llevan a mostrar un comportamiento basado en componentes. Sin embargo, los GameObjects son objetos y los componentes también. En particular, la implementación de los componentes son scripts que heredan de una clase MonoBehaviours, de la cual su ejecución está orquestada por los mecanismos internos de Unity que hacen que determinados métodos sean llamados en distintas fases del *loop* del juego.

En este capítulo presentaremos el modelado de las estructuras del sistema utilizando diagramas de clases UML. En el esquema empleado los GameObjects están asociados a objetos. Los hijos de los GameObjects, dentro de la jerarquía en escena de Unity, son relaciones de agregación. Los componentes de los GameObjects están diagramados como relaciones de composición. Usaremos letras itálicas para señalar nombres de assets o instancias de objetos en escena, los cuales pueden ser directamente rastreados utilizando el buscador de Unity.

6.2 Elementos de escena third-party

Montura de cámaras para visión de realidad virtual

La conjunción de los controles de los distintos dispositivos en la escena de Unity se implementa en una jerarquía de objetos a partir del modelo de montura de cámaras *LMHeadMountedRig*. La función principal de esta jerarquía de objetos es alinear la locación de las cámaras utilizadas por el casco virtual con la posición de las manos pro-

vistas por Leap Motion. A su vez, el controlador de Leap Motion mantiene un *factory* de manos a partir del cual se realiza la extensión para soportar manos con cualidades hápticas.

El asset principal de Leap Motion es el *LeapHandController* el cual se coloca en la escena y realiza el trabajo de ubicar las manos en el espacio virtual de Unity. Que incluye la administración de tipos, la conversión de unidades y de coordenadas. El *LeapHandController* actúa de ensamblador, usando el componente *HandPool* como factoría para crear representaciones virtuales de las manos. Las manos producidas son objetos que implementan la interfaz *HandRepresentation* y son una combinación de un modelo de mano 3D y un conjunto de scripts para coordinar los datos provistos por el controlador de Leap Motion los cuales designan la posición y orientación de todos los componentes de la mano en tiempo real.

OCULUS EN LA ESCENA

Para la visualización de la escena en Oculus Rift se utilizan dos cámaras, dadas por componentes *Camera* de Unity que se ubican dentro de los objetos *LeftEyeAnchor* y *RightEyeAnchor*. Las cuales generan la representación visual en cada uno de los ojos del casco virtual. Dado el soporte nativo de Unity a realidad virtual la conexión con el Oculus Rift se establece automáticamente y en el componente *Camera* se configura la asociación con el ojo humano al que corresponde, tanto los valores de separación de los ojos como la convergencia estereoscópica. Varios aspectos de ajustes del *rendering* se hacen automáticamente para adecuar la matriz de proyección y el campo de visión a las características soportadas por el casco.

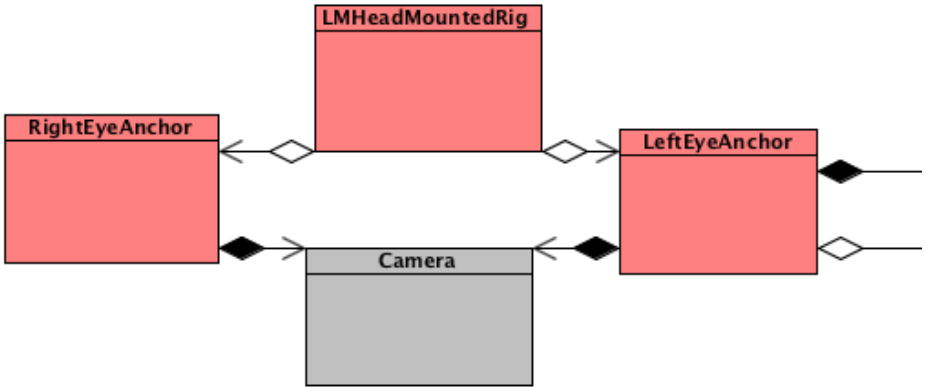


Figura 11: Componentes de Oculus en Unity.

LEAP MOTION EN LA ESCENA

Los componentes principales del controlador de Leap Motion se ubican en un objeto llamado *LeapSpace* dentro de uno de los objetos que representan la cámara. Es decir, el espacio operativo de las manos se ancla al campo de visión del casco virtual. Para alinear espacialmente la cámara y sincronizar las imágenes se introduce el componente *LeapVRTemporalWarping* que resuelve problemas de retraso en la sincronización de las manos que causan la sensación de desfase.

Dentro del *LeapSpace* se encuentra el prefab *LeapHandController* donde se ubican los componentes principales de operación con el servicio de Leap Motion. Estos son, *LeapProvider* que recibe los *frames* desde el servicio, a través de librerías en C de alto rendimiento los cuales traduce a clases de alto nivel en C# haciéndolas funcionales al resto de los scripts de Unity. También está el componente *HandPool,* que mantiene un *pool* de manos, donde se especifican distintos modelos para representan las manos que son activados y desactivados a medida que aparecen manos en escena. En nuestro caso utilizamos una representación de manos llamadas *HapticHands* las cuales proveen las utilidades para el diseño de las interacciones hápticas. Por último, el componente *LeapHandController* actúa de intermediario

entre el *LeapProvider* y el *HandPool*. Es decir, adquiere representaciones de manos del *HandPool* y las conduce utilizando los datos del *frame* adquiridos desde *LeapProvider*.

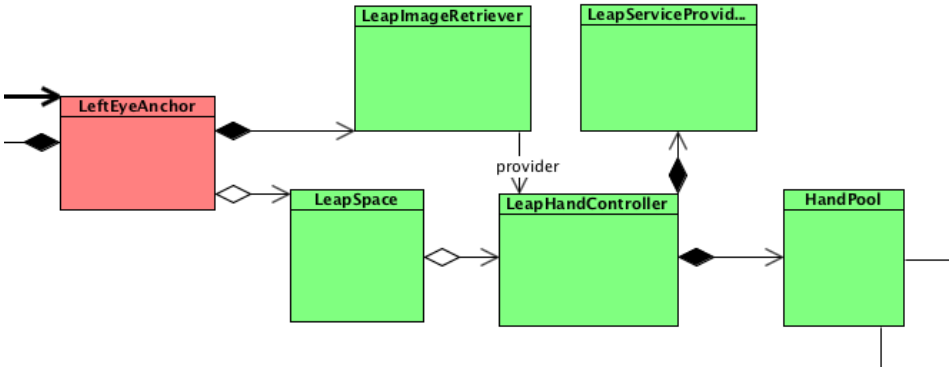


Figura 12: Componentes de Leap Motion en Unity.

6.3 MODELO DE DATOS DE MANOS VIRTUALES

El software de Leap Motion detecta manos y dedos dentro de su área de cobertura. Reportando sus posiciones, orientaciones, un conjunto de gestos y movimientos en *frames* a una tasa de refresco de entre 20 a 200 cuadros por segundo. Un *Frame* es una clase que describe un conjunto de datos recolectados en un momento específico.

El modelo de datos para las manos incluye un reporte de las características físicas de las manos detectadas. Los descriptores de las manos están encapsulados en la clase *Hand* las cuales mantienen una estructura con una lista de sus dedos en una estructura de clase *Finger*, también contiene descriptores de la palma y de su antebrazo. Cada *Finger* se corresponde ya sea con el pulgar, índice, medio, anular o meñique. A su vez cada *Finger* está compuesto por sus huesos, descriptos en la clase *Bone*.

En las utilidades del modelo de manos de Leap Motion se tiene una interfaz común para todas las manos en la clase abstracta *IHandModel* en donde se define la quiralidad de la mano instanciada, si es la izquierda o la derecha; y en donde se configura el tipo de modelo, si es gráfico o físico.

Por cada mano real se puede trasladar en Unity un modelo gráfico o físico de la misma. En el modelo gráfico se determina como se visualiza la mano en el mundo virtual. En el modelo físico en cambio se configura como interactúa físicamente con los objetos de la escena, a través de una configuración de colisionadores físicos.

Para crear una mano en Unity se debe implementar *IHandModel*. En los assets provisto por Leap Motion se incluyen las siguientes clases que extienden *IHandModel* y permiten utilizar distintos modelos de manos en una escena de Unity.

- *RigidHand*: Un modelo físico para las manos compuesto de varios *Collider*.
- *RiggedHand*: Una representación gráfica de la mano basada un modelo 3D usando articulaciones.
- *DebugHand*: Un modelo que dibuja líneas para los huesos en la mano y sus dedos. Se activa solo en el editor.

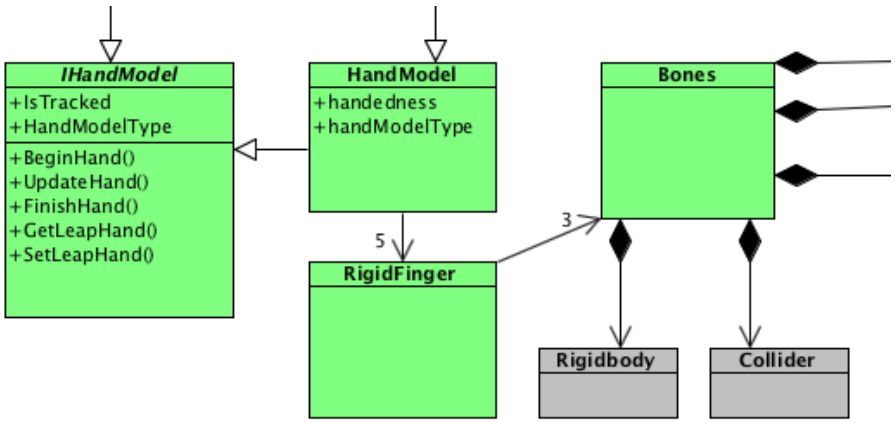


Figura 13: Modelo de datos de representación de manos en Leap Motion.

6.4 MODELO HÁPTICO

Para la implementación de la herramienta háptica se consideraron los siguientes objetivos:

- La herramienta se enfoca en el diseñador. El flujo de trabajo está centrado en que no haya que escribir código y que se pueda implementar y probar rápidamente las interacciones que se diseñen.
- Se utilizan las herramientas de Unity para extenderlas al diseño de interacciones hápticas. El usuario que está acostumbrado a trabajar en el editor encuentra intuitivo el flujo de trabajo propuesto.
- La herramienta háptica debe permitir diseñar y probar los patrones usando Unity tanto en modo editor como en el reproductor.

6.5 MANOS HÁPTICAS

La herramienta de diseño háptico implementa dos representaciones de manos virtuales con el propósito específico de dotarlas de cualidades hápticas, una para el modelo gráfico y otra para el físico. Estas son una extensión de las manos del conjunto de utilidades de Leap Motion, las cuales son provistas por funciones para la interacción con materiales hápticos. La instanciación de los modelos ocurre en el *HandPool* donde se configuran los prefabs asociados para las manos izquierda y derecha correspondiente a cada uno de los dos modelos.

6.5.1 MODELO HÁPTICO FÍSICO

El modelo físico llamado *HapticPhysicHand* es una extensión del prefab *RigidRoundHand*, el cual consiste en un modelo rígido compuesto de varios *Colliders*. La estructura contiene *Colliders* para el antebrazo, la palma y los cinco dedos. Las formas geométricas que los definen son: una caja para la palma y para cada uno de los huesos de las manos se utiliza una geometría de cápsula, dando una precisión eficiente de colisiones sin consumir demasiados recursos. El propósito original de esta estructura es la de integrar las manos con el motor de física de Unity para que el usuario interactúe con los objetos de escena produciendo colisiones. Para fines de empleo en háptica esta estructura se aprovecha principalmente para detectar interacciones con materiales hápticos y para configurar la distribución de actuadores.

El comportamiento físico de las manos de Leap Motion es que pueden interactuar con los objetos de la escena virtual provocando reacciones físicas en ellos, pero sin que repercutan en el modelo físico de la

mano. Esto se debe a que, para mantener la sensación de presencia, las posición y orientación de las manos recreadas en el mundo virtual no pueden ser discordantes con las manos reales. Por esa razón los *Colliders* que componen el modelo físico son del tipo *kinematic rigidbody colliders*.

La interacción de materiales consiste en los componentes de detección de colisiones con materiales hápticos y el detector de *triggers* con volúmenes de zonas hápticas. Funciones provistas por los componentes *HandTriggerDetector* y *HapticMaterialDetector*. Estos componentes realizan la tarea en tiempo de ejecución de extender los objetos de los huesos con detectores los cuales son administrados por las manos. Ante la ocurrencia de un evento de interacción háptico disparan un determinado estímulo previamente diseñado en el editor.

La distribución de los actuadores se realiza en el editor de la escena de un modo intuitivo. Para cada hueso de los dedos, es decir, las falanges distales; medias y proximales, puede o no haber un actuador. El diseñador determina a cuál de ellos asignarle un actuador simplemente asignándole un componente *Actuator*. Los nombres de los dedos y cada uno de sus huesos se visualiza en la jerarquía de objetos de la escena para facilitar su ubicación. A esto se le agrega una ayuda visual de la mano indicando cual tiene un actuador y cual no, la cual es provista en el modelo gráfico.

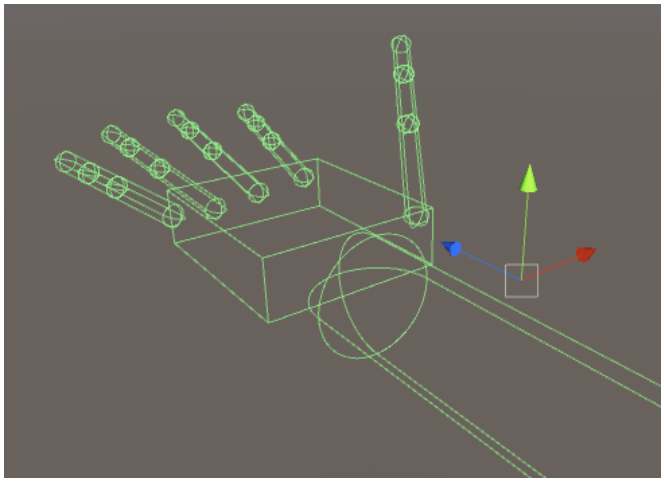


Figura 14: Conjunto de colisionadores que componen el modelo físico.

6.5.2 MODELO HÁPTICO GRÁFICO

El modelo gráfico llamado *HapticGraphicHand* se implementó con el objetivo de reflejar visualmente los estímulos producidos a los actuadores. Por lo tanto, el propósito es el de asistir el diseño de los patrones hápticos y en la realización de pruebas. Como no tiene funciones de detección de interacciones ni representaciones de estímulos, el prefab puede ser reemplazado por el modelo gráfico preferido en producción o luego de terminar las pruebas. Las funciones que provee, y que funcionan de soporte al diseño háptico, son el de indicar visualmente en donde están ubicados los actuadores y la visualización de los estímulos en cada hueso del dedo mediante una representación en un rango de colores.

El refuerzo visual del modelo gráfico al diseño de las interacciones hápticas se sustenta en que las manos son objetos persistentes en el editor de Unity. En modo editor, que es donde transcurre el trabajo de diseño de interacciones, las manos adoptan una pose por defecto y sus componentes son refrescados continuamente.

La implementación es una extensión del prefab *CapsuleHand* el cual es provisto originalmente como ejemplo de un modelo gráfico construido dinámicamente, a diferencia de uno usando geometría preexistente como pueden ser las mallas poligonales. La cualidad de generar geometrías dinámicamente abre la posibilidad de cambiar al vuelo las formas y colores del modelo para representar los cambios en los estímulos. Si bien los modelos gráficos y físicos suelen guardan independencia entre sí, en nuestra implementación el modelo gráfico referencia al físico para obtener el valor de los actuadores en cada *frame*.

La representación visual de la mano se lleva a cabo mostrando solo geometrías de esferas y cilindros. Las terminaciones como los nudillos y otros extremos son representadas como esferas negras. El resto de los huesos se representa con cilindros, los cuales indican la presencia de un actuador cuando aparecen en blanco o la ausencia de los mismo cuando están grises.

Para reproducir los estímulos visualmente, en cada *frame* el modelo gráfico comprueba el estado de los actuadores. Por cada hueso afectado muestra, en un rango lineal discreto, una mezcla del blanco al rojo para los estímulos de un voltaje nominal del mínimo al máximo de los actuadores.

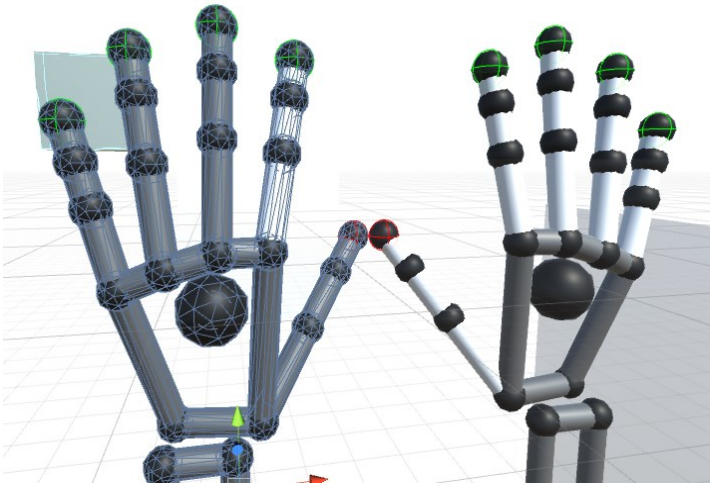


Figura 15: Representación en escena de un modelo gráfico con actuadores en el dedo índice de la mano izquierda y la mano derecha completa.

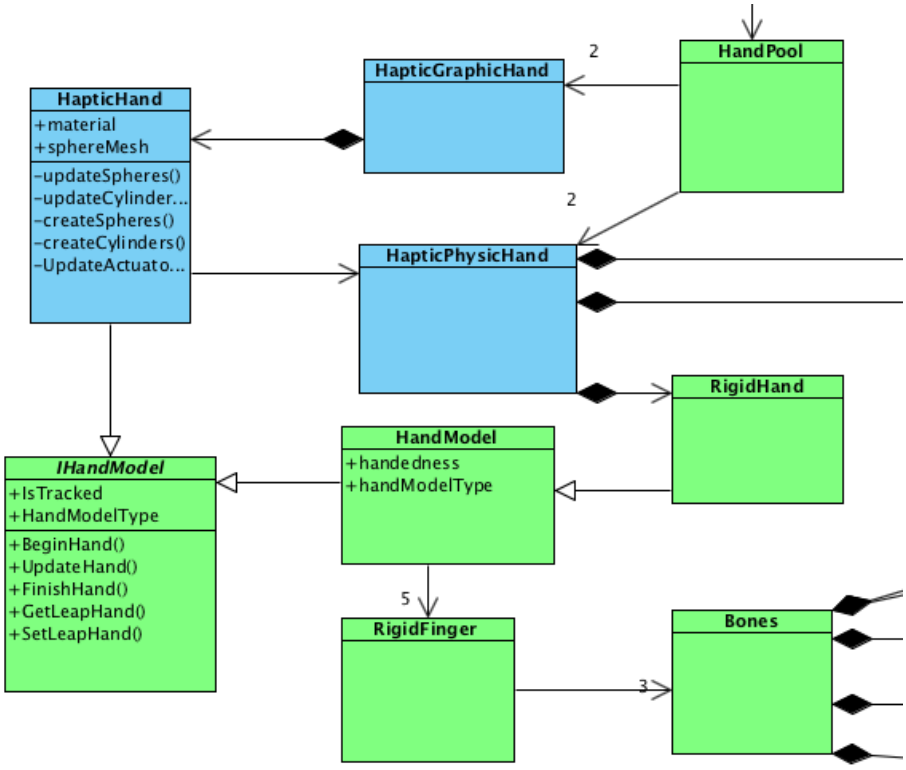


Figura 16: Modelo de datos háptico en composición con Leap Motion.

6.5.3 Actuadores

Los actuadores físicos del dispositivo háptico se colocan en la parte frontal de los dedos correspondientes a los huesos de los dedos. Estas tres zonas, que comprenden la yema de los dedos y la parte frontal de los restantes huesos de la falange, son las partes más sensibles de la mano y es donde se optimizan los resultados obtenibles del output háptico.

El actuador del dispositivo háptico es representado como un componente de Unity que se integra en un hueso del modelo físico de la mano de Leap Motion con el nombre *Actuator*. Esto es, en los objetos terminales de la jerarquía de elementos del modelo físico de la mano. La asociación al modelo físico facilita la tarea del diseñador de localizar el hueso correspondiente a través de la jerarquía de objetos. A nivel funcional, el hueso del modelo físico como tal contiene un *Collider* lo que le permite al actuador escuchar los eventos de colisiones con otros objetos de la escena y disparar las interacciones hápticas en colaboración con detectores de eventos asociados a la mano.

La función principal del *Actuator* es procesar el valor del voltaje de la señal que se emite al controlador electrónico para producir la sensación de tacto. El valor se restringe a los valores de un flotante normalizado entre 0 y 1. Donde el valor 0 indica la ausencia de señal y los valores positivos son usados para provocar la aceleración de la fuerza del actuador. En segundo lugar, los componentes *Actuator* se configuran en el inspector del editor indicando el número de pin de salida de la placa Arduino donde se refleja la correspondencia con su respectivo motor físico.

En el inspector se muestra un color para dar una respuesta visual al diseñador de la intensidad de la señal que se está procesando, de manera similar a como se muestra la mano en colores en la ventana de la escena.

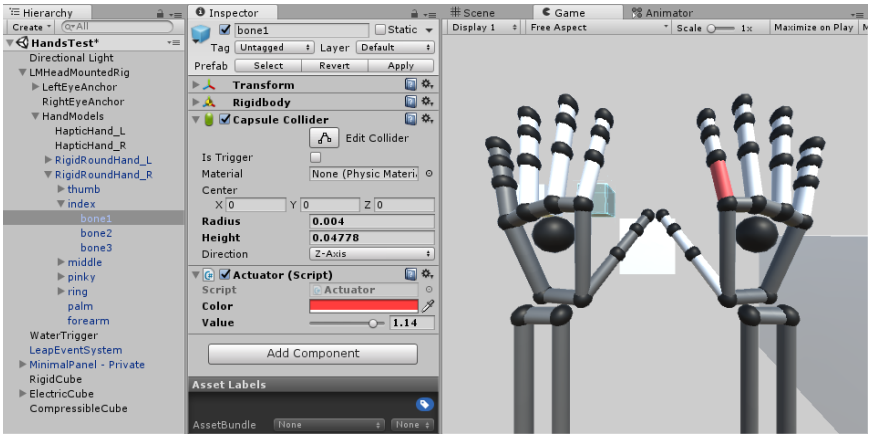


Figura 17: Jerarquía del modelo de la mano, edición del actuador y su representación visual en el editor.

Como ayuda visual en la configuración de la disposición de actuadores de la mano, los huesos se pintan en gris en caso de ausencia de actuador, o de blanco en caso de presencia.

6.6 Controlador háptico

El controlador háptico es la parte de la herramienta que se encarga de la etapa de *rendering* háptico, donde recibe información de los *Actuators* y refleja el estado en los motores físicos mediante la comunicación con el dispositivo háptico.

La interfaz entre la herramienta y el dispositivo háptico ocurre en un objeto llamado *HapticDeviceController*. El cual actúa de forma pasiva en la escena recibiendo mensajes de los *Actuator* en forma de actualizaciones ante cualquier cambio de los valores del actuador. El valor normalizado de los *Actuator* entre 0 y 1 es transformado a la frecuencia de PWM del rango del byte, es decir entre 0 y 255, donde 0 significa siempre apago y 255 significa siempre prendido. La configura-

ción de este componente se realiza indicando en el inspector el nombre del puerto serie al que se encuentra conectada la placa Arduino y, opcionalmente, el *baud rate* que designa la cantidad de bits por segundos, donde por defecto del estándar Firmata toma el valor 57600.

La comunicación con la placa Arduino se realiza a través de un puerto serie utilizando el protocolo de comunicación Firmata. Las funciones de puertos series en Unity son provistas por las librerías de clases de la API de .Net en la versión 2.0. La configuración para que el puerto serie funcione son los datos que se definen en *HapticDeviceController* descriptos anteriormente. El protocolo Firmata permite operar la placa Arduino desde el dispositivo host donde se ejecute la aplicación o el mismo editor Unity. La forma de interactuar con Arduino se basa fundamentalmente en el envío de mensajes analógicos, es decir los que utilizan el módulo PWM para simular rangos de frecuencia analógicos. Estos mensajes se envían desde Unity a la placa Arduino cada vez que un *Actuator* modifica su valor, indicando el pin al que corresponde el motor y el nuevo valor que designará la fuerza vibrotáctil que recibe el usuario en ese momento.

6.7 INTERACCIONES HÁPTICAS

6.7.1 MATERIALES

Los materiales hápticos son componentes que describen el efecto sensitivo que dispara en la mano cuando entra en contacto.

Los materiales van asociados siempre a objetos con *Colliders* del tipo *rigidbody collider*, denominación que se emplea para describir

cuando el objeto tiene asociado un *Collider* que no es *trigger*[10] y un *Rigidbody* que no es *kinematic*. Es por eso que los objetos interactuables siempre tienen una reacción física. Las manos no pueden tener reacciones físicas al entrar en contacto porque de esta manera se rompería la sensación de presencia de las manos reales en el mundo virtual, en cambio solo pueden provocar reacciones en los objetos con los que colisiona.

Tener la descripción del material en el objeto mismo que lo representa permite diseñar directamente sobre el objeto que se trabaja y desacopla la definición de las manos y el resto de los componentes hápticos.

Las curvas que expone el inspector del editor son editadas por el diseñador en el editor de curvas de Unity. Las curvas son modificadas agregando y quitando llaves en el eje horizontal y actuando sobre las tangentes. Las curvas se pueden guardar y ser reutilizadas mediante la librería de curvas predeterminadas. Con la reutilización de curvas los efectos se diseñan y se prueben una única vez para posteriormente ser replicados en otros objetos con facilidad.

[10] Un trigger no registra colisiones cuando entra en contacto con un Rigidbody. En cambio, envía señales cuando el Rigidbody entra y sale del volumen, sin desencadenar reacciones físicas.

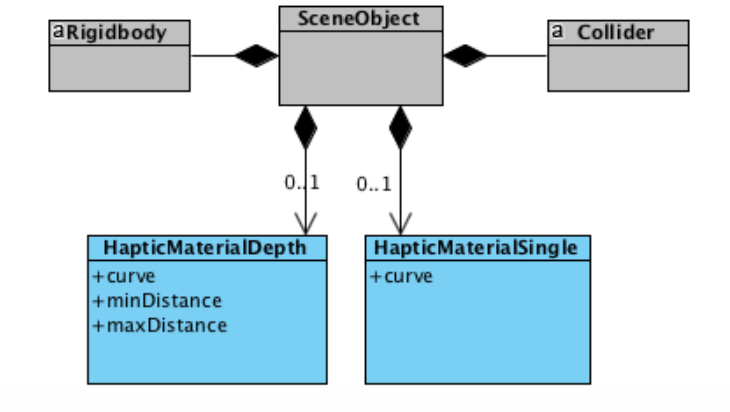


Figura 18: Estructura de los descriptores de materiales para objetos de escena.

Los componentes que definen los materiales de los dos tipos de interacciones táctiles contempladas son:

- *HapticMaterialSingle*: contiene una curva que define el efecto en el hueso que entra en colisión con el material. El efecto se dispara por única vez en la fase inicial de colisión, cuando entra en contacto. Ignorando las siguientes etapas de colisión hasta que el efecto termine y vuelva a producirse el contacto. La curva define una línea de tiempo con los valores que asume el actuador. Donde la duración total, dada en unidades de segundos, es la distancia de la primer a la última llave.
- *HapticMaterialDepth*: contiene una curva que define los valores del actuador en relación a la profundidad de penetración de la mano en el objeto interactuable. En este caso la curva define una función en donde la variable horizontal es el grado de penetración. El eje vertical designa el valor del actuador del mismo modo que en *HapticMaterialSingle*. La parametrización del componente está configurada por los campos de distancia máxima y mínima en metros que son contemplados en los contactos de la mano y el colisionador. En el motor de física la colisión comienza cuando los puntos de contactos alcanzan una

distancia menor a la suma de los desplazamientos de contacto entre un par de colisionadores. Por lo tanto, la distancia puede ser positiva, cuando no están interpuestos; puede ser cero cuando están en contacto; o positiva cuando hay una interposición. Este rango de distancias sirve para describir diferentes texturas en objetos rígidos.

Figura 19: Componente para definir materiales simples.

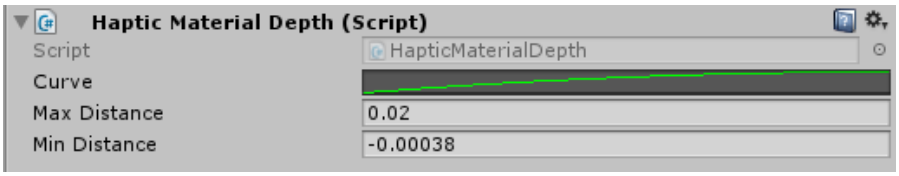


Figura 20: Componente para definir materiales por profundidad.

6.7.2 Diseño de patrones hápticos

La herramienta está compuesta de tres modos de diseñar patrones hápticos, cada uno representado un modo de interacción distinto. Se diseñan en dos partes, un componente se encarga de detectar la ocurrencia del evento disparador y en otro componente se define la reacción háptica.

6.7.2.1 Estímulo sincrónico

El primer modo consiste en el diseño de efectos sincrónicos en varios actuadores de la mano. Útil para diseñar efectos desencadenantes y efectos en los que es preciso orquestar los actuadores en simultaneo.

Los escenarios donde se aplica es cuando hay contacto con un volumen no rígido como puede ser fuego o agua, cuando se recrea un efecto involuntario como puede ser un escalofrío, o cuando se simula una reacción compleja como puede ser la caída de un objeto de la mano.

Los componentes que lo describen se asocian a la mano, tanto el detector como el descriptor del efecto. El componente *HandTriggerDetector* se encarga de detectar cuando la mano entra en contacto con un *Collider* configurado como *trigger*. En este caso no sucede colisión física y solo se notifica la ocurrencia del ingreso de la mano dentro de un volumen. El componente es configurado en sus parámetros públicos para indicar que animaciones desencadenar.

El estímulo se diseña usando el sistema de animaciones de Unity. El sistema de animaciones, también referido como Mecanim, provee un flujo de trabajo para configurar animaciones en todos los elementos de Unity. En la herramienta háptica se trabaja sobre las propiedades de los actuadores. Consiste en un conjunto de herramientas visuales para previsualización de clips de animación y transiciones, que de otro modo se configurarían en programación.

El flujo de trabajo para los estímulos hápticos consiste en asignar un *AnimationClip* por cada efecto a recrear en el controlador de animación. El efecto a disparar es informado desde el detector al animador al reconocer el objeto de interacción cuando se ingresa al volumen de colisión. El *AnimationClip* tiene una curva a configurar por cada actuador previamente seleccionado. La herramienta visual permite la edición simultánea de todas las curvas involucradas con lo cual se diseñan efectos sincrónicos. Las curvas definen el valor del actuador en una línea de tiempo. La animación se puede configurar en *loop* o como un estímulo de una sola ejecución.

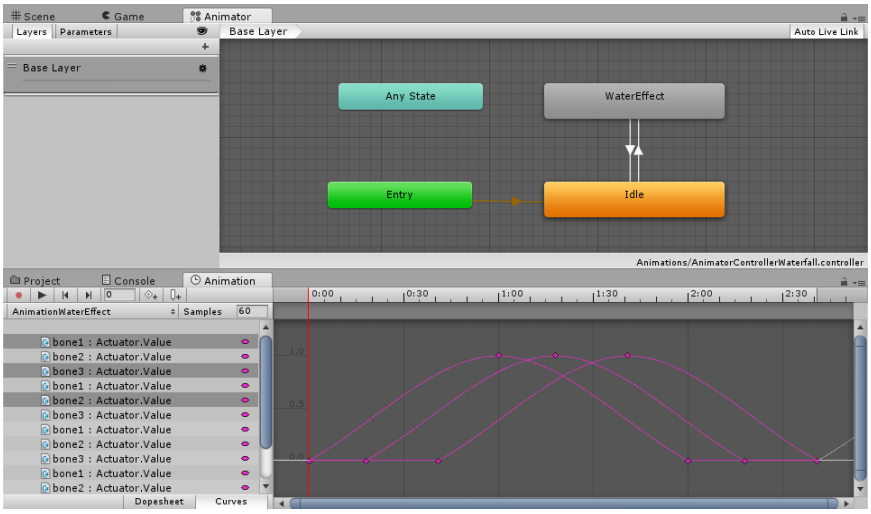


Figura 21: Ventanas de diseño del sistema de animaciones.

6.7.2.2 ESTÍMULO DE CONTACTO CON MATERIAL SIMPLE

Este estímulo es disparado cuando una parte de la mano asociada a un actuador entra en contacto con un material háptico y representa el modelo de interacción más básico. El estímulo consiste en un efecto que se reproduce en el actuador en un tiempo determinado. Tales interacciones tienen lugar cuando el contacto con el material háptico sucede en un lapso pequeño de tiempo y no se mantiene el contacto, luego del cual el objeto interactuable se separa rápidamente o desaparece. Algunos casos de interacción pueden ser el contacto con elementos eléctricos o la recolección de un ítem el cual desaparece al contacto.

La detección de materiales de contacto se realiza en el componente *HapticMaterialDetector* incluido en la mano háptica. Cuando se produce una colisión en uno de los huesos asociados a un actuador se comprueba si el colisionador tiene propiedades hápticas en cuyo caso se dispara el efecto. El componente que describe en este caso el efecto a reproducir para el material es *HapticMaterialSingle* descripto en la sección de materiales hápticos.

6.7.2.3 ESTÍMULO DE CONTACTO CON MATERIAL POR PROFUNDIDAD

Las interacciones de contacto por profundidad ocurren, al igual que las de contacto simple, cuando una parte de la mano asociada a un actuador entra en contacto con un material háptico. La diferencia fundamental es que estos contactos se describen según el nivel de penetración de la mano en los objetos con materiales de estas características. Otra diferencia fundamental es que no es un efecto que sea disparado por un evento una única vez y se describa por una línea de tiempo, sino que la interacción transcurre por todo el período de tiempo que dura el contacto entre los colisionadores. Y a diferencia de la línea de tiempo, el efecto se describe por el grado de penetración de la mano en el objeto. Esta interacción es útil para diseñar estímulos que describen materiales complejos, cuerpos rígidos y maleables, o diferentes texturas.

La detección del material ocurre, al igual que las interacciones de contacto simple, en los componentes *BoneTouchDetector* asociados al *HapticMaterialDetector*. En este caso la interacción se observa al comienzo de la colisión, perdura durante la continuidad del contacto y se da por finalizada cuando la colisión termina. El estímulo resultante se define en el componente *HapticMaterialSingle* asociado al objeto con propiedades hápticas que colisiona con la mano. En el componente háptico se diseña la curva como representación de una función, en la que el dominio identifica el grado de penetración de la mano y el codominio refleja el estímulo asignado al actuador.

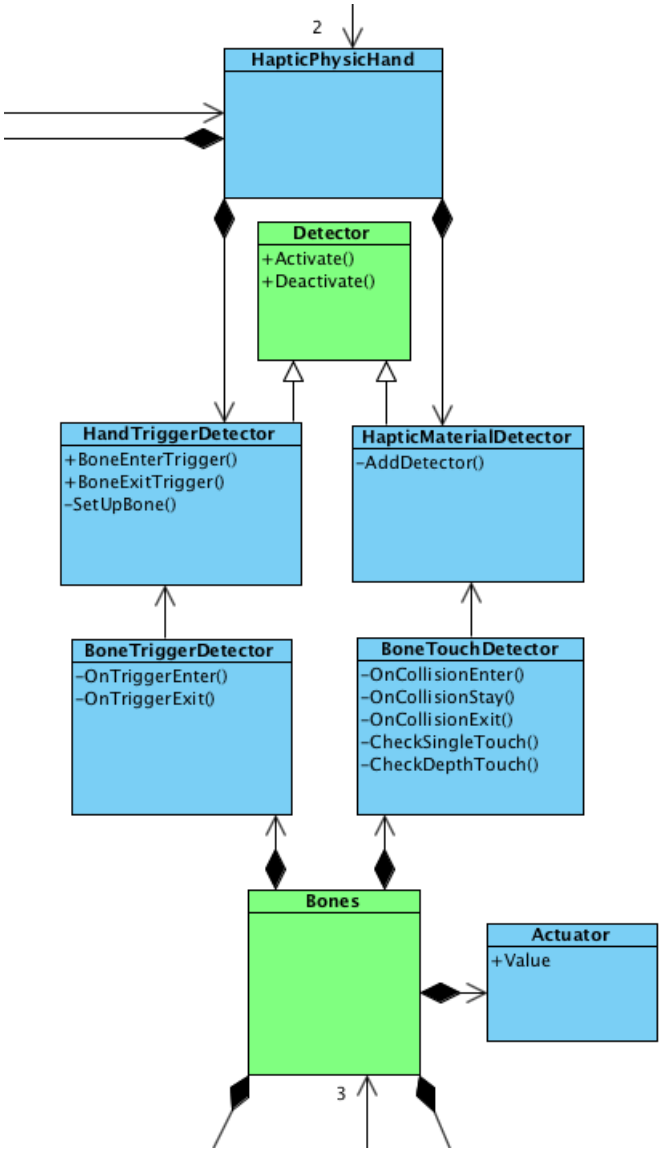


Figura 22: Estructura de los detectores que desencadenan las interacciones hápticas.

Por último, en el siguiente diagrama mostramos los eventos que ocurren al detectarse la interacción de la mano con un objeto de cualidades hápticas con material por profundidad. En el modo de interacción por contacto simple, la secuencia de eventos que se desencadena es de características similares.

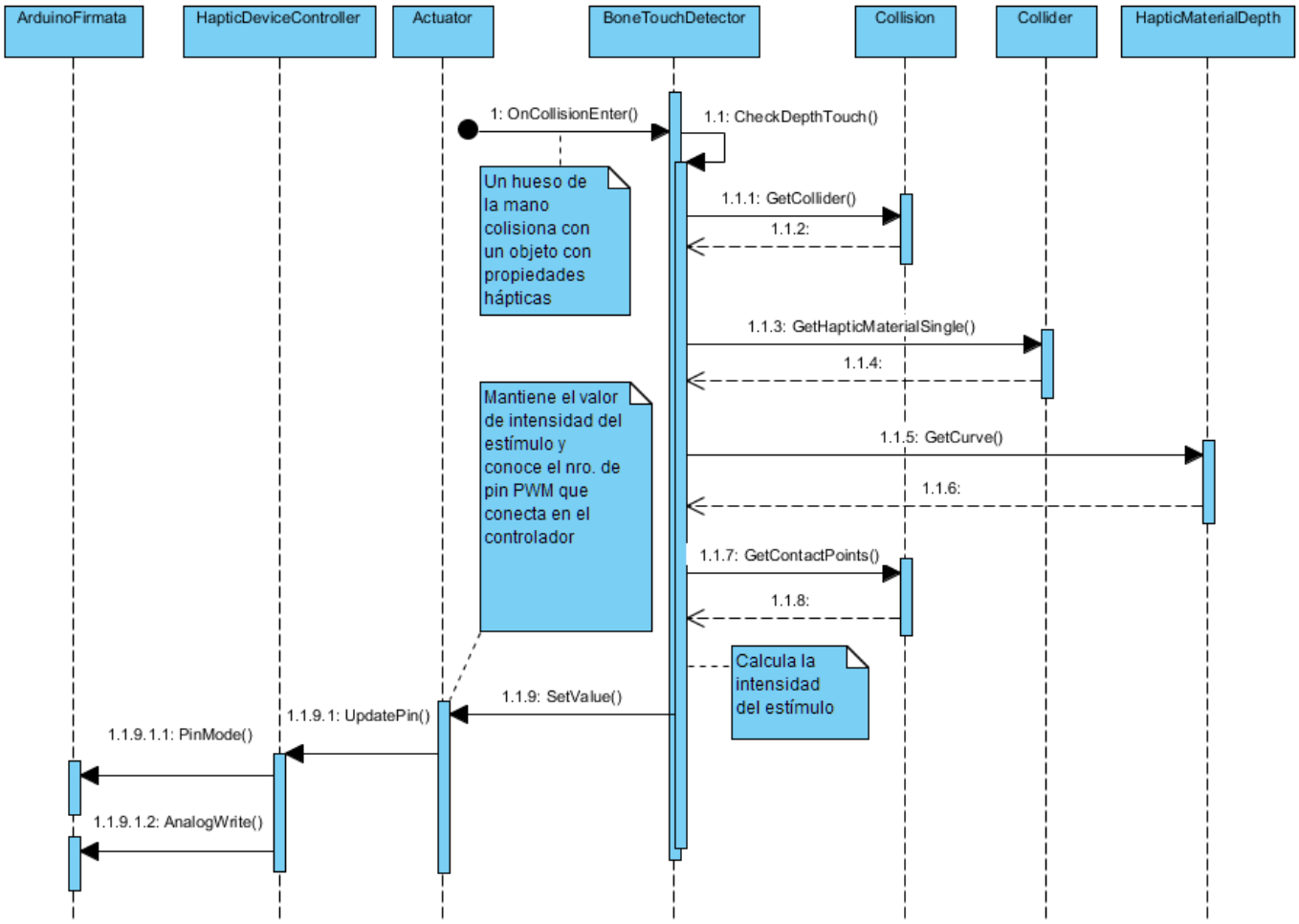


Figura 23: Diagrama de secuencia, detección de un material y procesamiento de la señal.

7 EVALUACIÓN

En este capítulo describiremos pruebas paso a paso de utilización de la herramienta a fin de demostrar el modo de emplearla y examinar algunos ejemplos de las interacciones que se pueden llevar a cabo para describir estímulos hápticos en un entorno de realidad virtual.

Como antecedente de las pruebas conceptuales sobre la herramienta y a modo de comprender la forma de utilizarla, comenzamos esta sección con una introducción para entender cómo está organizado el proyecto. Luego se describe cómo se establece una configuración inicial típica para el entorno virtual con háptica en Unity.

Por último, se describen una serie de pruebas conceptuales sobre diferentes modos de generar interacción táctil con objetos virtuales. Estas pruebas son concebidas, en primera instancia, como una forma de evaluar la herramienta mediante ejemplos concretos de cómo se implementan las utilidades provistas, y al mismo tiempo, como un modo informativo sobre las distintas formas en que puede ser utilizada.

Es importante destacar que existe una gran variedad de técnicas para implementar interacciones hápticas ya que no están predeterminadas en la herramienta. En los ejemplos mostrados en esta sección se intenta demostrar una de las formas más sencillas de generar y describir interacciones mediante la reutilización de componentes.

7.1 INTRODUCCIÓN A LA HERRAMIENTA

Las herramientas hápticas consisten en un conjunto de assets en Unity que proveen utilidades para dotar las escenas con capacidades

hápticas, diseñar estímulos y establecer las interacciones que las desencadenan.

IMPORTACIÓN Y DEPENDENCIAS

Para que funcionen las utilidades de la herramienta de autoría es necesario que previamente esté incorporada en el proyecto la librería principal de Leap Motion llamada Unity Core Assets Orion Beta. La dependencia con esta librería está dada principalmente por la utilización de scripts en composición de objetos y herencia en el caso de la reimplementación de las manos virtuales para háptica. Leap Motion también provee módulos que extienden las funcionalidades del Core Assets los cuales son opcionales y no son dependencia de la herramienta háptica.

Es importante notar que los assets de Leap Motion están provistos de manera experimental y que en sucesivas versiones han ido cambiando las interfaces de sus clases, es decir, no tienen una API establecida. Por lo tanto, es posible que nuevas versiones de estos assets requieran actualizaciones de las utilidades de la herramienta háptica o la dejen obsoleta. La versión de la librería que es soportada actualmente es la 4.1.6.

Después de importar los assets de Leap Motion se incorporan los assets de la herramienta de autoría háptica mediante la importación de un *unity package* provisto como parte de esta tesina.

ENLACES DE INTERÉS

En esta sección listamos los enlaces para acceder a las dependencias del proyecto.

https://github.com/miltonloy/haptic-tools-unity

Herramienta de autoría háptica para Unity

https://github.com/leapmotion/UnityModules/releases/tag/Release-CoreAsset-4.1.6

Leap Motion Core Assets

https://unity3d.com/es/get-unity/download/archive

Unity 5.5.3

https://developer.leapmotion.com/windows-vr

Leap Motion Orion Runtime

https://www.oculus.com/setup/

Oculus Rift Setup

ESTRUCTURA DE DIRECTORIOS

La estructura de directorios del proyecto queda conformada por la carpeta *HapticTools*, que contiene todos los assets de la herramienta háptica. Estos assets trabajan conjuntamente con las herramientas para Unity provistas por Leap Motion, los cuales pueden ser importados o actualizados de manera independiente.

La estructura de directorios de la herramienta háptica queda conformada de la siguiente manera:

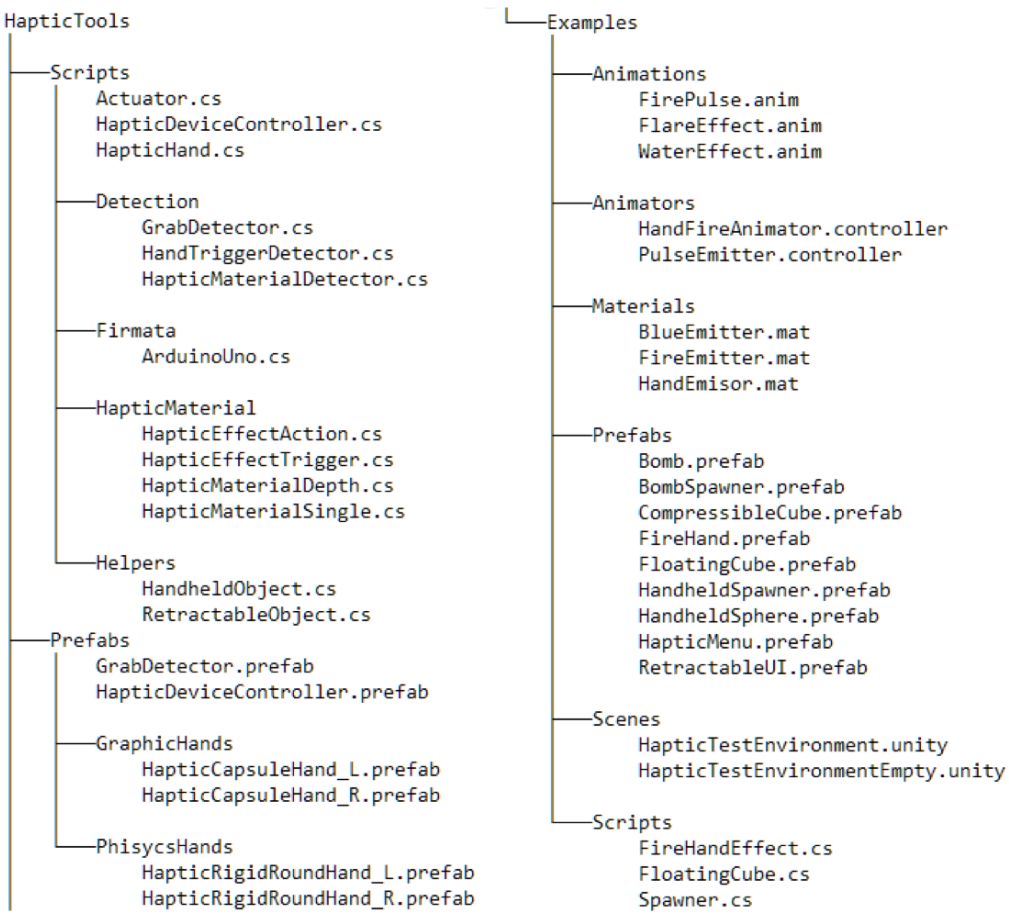


Figura 24: Vista en árbol de la estructura de ficheros de la herramienta.

En la carpeta Scripts están los principales *MonoBehaviours* que componen la herramienta. Por ejemplo, los *Actuators* que se arrojan directamente sobre los huesos para indicar la presencia de un actuador físico y configurar el enlace. En el caso de los scripts de la carpeta *HapticMaterial* hay diferentes componentes para arrojar sobre objetos de escena y convertirlos en objetos hápticos que describen distintos tipos de interacción. En la carpeta *Prefabs* están los principales elementos para configurar de manera inicial una escena con integración háptica. Por último, en la carpeta *Examples* se encuentran una

gran variedad de componentes que han sido desarrollados para probar conceptos de interacción y los cuales describen buena parte de los conceptos que explicamos en esta sección.

7.2 CONFIGURACIÓN INICIAL DEL ENTORNO HÁPTICO EN UNITY

La configuración del entorno háptico puede ser descripta en tres pasos fundamentales: la incorporación de la representación de manos virtuales en escena, la dotación de cualidades hápticas a las manos, la configuración del enlace con el dispositivo háptico y los actuadores.

El primer paso es agregar una instancia del prefab *LMHeadMountedRig* de la librería Leap Motion a escena. Este prefab contiene el *LeapHandController* que enlaza el dispositivo de reconocimiento de gestos, además mantiene el *pool* de manos virtuales y provee una cámara con cualidades estereoscópicas para ser utilizado en realidad virtual con el reconocimiento de manos orientado de manera frontal a las manos. Para que funcione la cámara en realidad virtual se debe marcar la opción *Virtual Reality Supported* en el PlayerSettings del editor de Unity.

El siguiente paso es cambiar la representación de manos virtuales de Leap Motion. La herramienta háptica tiene como requisito un modelo físico de manos basado en representaciones de huesos basados en colisionadores. Con este fin se puede utilizar un modelo como el *HapticPhysicHand* u otros de características similares como los provistos por el Core Assets de Leap Motion. Este prefab es, en esencia, una representación física de Leap Motion con un componente de detección de materiales hápticos. El componente *HapticMaterialDetector* es el distintivo fundamental que posibilita la detección de materiales táctiles y debe ser incorporado en caso de utilizar otra representación que no sea la provista por la herramienta háptica. En cualquier caso,

solo no serían compatibles aquellos modelos que no usan colisionadores o que los generan dinámicamente.

Para el modelo gráfico se provee el prefab *HapticGraphicTestHand* el cual es concebido como una mano para ser usada durante el desarrollo donde se puede testear la ubicación de los actuadores y el feedback háptico de una manera visual. En caso de utilizarse el modelo físico háptico, se debe generar una instancia y asociarle la referencia del modelo físico, ya que esta representación de manos utiliza la información de los contactos físicos generados para cambiar la visualización del modelo gráfico.

Una vez definidas e instanciadas las representaciones de las manos, hay que configurar el *HandPool* del *LeapHandController* incluyendo las referencias a cada una de las representaciones de las manos, con un grupo para el conjunto de manos gráficas y uno para el físico.

Una vez instanciado el modelo de manos virtuales y vista estereoscópica se configura el dispositivo háptico. Para esto, se instancia en escena un *HapticDeviceController* y se configura en el inspector el puerto serie donde va conectado el microcontrolador Arduino que tiene asociado los actuadores. Luego se configuran los actuadores agregando componentes *Actuator* a cada uno de los huesos del modelo físico de manos de la mano que corresponda.

Con esto queda completamente configurado el esquema del entorno de realidad virtual en Unity con interacciones hápticas. A partir de aquí el proceso de diseño se centra en la descripción de los estímulos hápticos y en la definición de las interacciones que los desencadenan.

7.3 Pruebas de concepto

7.3.1 Generando tacto en objetos virtuales

Un caso de aplicación sencillo e inmediato en la incorporación de háptica en escenas virtuales sería el caso de tocar objetos. A continuación, daremos una descripción de los procedimientos básicos para dotar de cualidades táctiles a objetos virtuales.

Un objeto en Unity para poder tener contacto físico con otros objetos debe tener por un lado un componente Rigidbody que es el encargado de calcular la física del objeto y por otro un componente Collider que describa el mesh de colisión que se va a utilizar. Luego se proporcionan las cualidades hápticas al objeto utilizando la herramienta de autoría.

A continuación, describiremos dos tipos hipotéticos de interacciones táctiles posibles.

Contacto prolongado por profundidad

Si quisiéramos tocar de modo prolongado un objeto, una aproximación sería plegar el objeto ante el contacto con la mano y retraerlo ante la ausencia de contacto para volverlo a su lugar. Con esto logramos que la mano no atraviese el objeto, ya que el feedback vibrotáctil no contempla sensaciones kinestésicas como las que podrían provocar que la mano se detenga al contacto con un objeto virtual rígido.

Una forma de conseguir esto es, en principio, agregando un componente *HapticMaterialDepth* al objeto. Realizando esto la mano ya recibe feedback háptico según el grado de penetración con el objeto al contacto. Pero la mano saldría volando de inmediato al producirse el contacto. Es por eso que, para retraer el objeto, se aumenta significativamente la fuerza de arrastre el cual está definido en la propiedad

drag en el *Rigidbody* para dificultar la energía necesaria para empujar el objeto. Por último, para retraer el objeto a la posición inicial se incorpora el componente helper de la herramienta llamado *RetractableObject*.

ESTÍMULO TÁCTIL AL CONTACTO MOMENTÁNEO

Otro caso supuesto sería el caso en que queramos describir un estímulo táctil de variada intensidad que dura un tiempo determinado y que se desencadena por un solo contacto con un objeto virtual.

En este caso el objeto tiene, como todo objeto de contacto, un Rigidbody y un Collider. Pero en este caso el Rigidbody debe poseer un drag bajo para que se mueva rápidamente al ser tocado. Luego el material háptico que describe este tipo de eventos es el llamado *HapticMaterialSingle*. Este componente tiene una propiedad que es una curva de animación la cual describe la duración del efecto en el eje horizontal y la intensidad de la señal en el eje vertical.

Con esta configuración al tocar el objeto con las manos este saldría disparado por los aires y en los huesos que entraron en contacto con el objeto se desencadena un estímulo durante un tiempo determinado.

7.3.2 ESTÍMULOS AL SUJETAR OBJETOS VIRTUALES

Un caso de estudio de interacción háptica es como representar en háptica la sujeción de un objeto virtual. Para esto vamos a suponer un objeto de interacción por contacto típico, como puede ser un objeto primitivo esfera compuesto por un Rigidbody y un Collider. Al incorporarle un *HapticMaterialDepth* el objeto adquiere propiedades hápticas.

El problema con que nos encontramos es que, al intentar agarrarlo, el motor de simulación física produce efectos indeseados como que el objeto sea atravesado o que salga disparado si se presiona contra otros colisionadores presentes en la escena. Esto es consecuencia de un modelo de movimiento libre de la mano, y los objetos virtuales que no tienen la misma libertad de movimiento que la mano y al entrar en contacto con otros colisionadores se generan inconsistencias.

Para eludir estos impedimentos, la herramienta provee un conjunto de helpers para generar un modelo de sujeción. Un modelo similar de simulación es provisto por el módulo de Leap Motion llamado InteractionEngine pero este no es compatible con el esquema de colisionadores requerido en la representación física de las manos virtuales de la herramienta. Los helpers de la herramienta háptica consisten en agregar una serie de detectores los cuales se integran en el prefab *GrabDetector* el cual debe instanciarse dentro de la jerarquía de las manos virtuales. Luego los objetos virtuales que tengan la cualidad de poder ser sujetados se asocian al *layer handheld* y se les agrega el componente *HandheldObject* el cual se encarga de interrumpir la simulación de física fijando el objeto a la traslación y rotación de la mano.

Una vez incorporados estos elementos el modelo de sujeción está en marcha y al momento de agarrar entre los dedos un objeto de estas características lo que sucederá es que el objeto quedará unido al pellizco de los dedos. La salvedad es que sacrificamos la simulación física de contacto para poder sujetar el objeto, donde los huesos de las manos posiblemente atraviesen algunas partes del objeto. Y el estímulo táctil de contacto continuará la simulación, pero intensificando la señal en los huesos que hayan penetrado el objeto.

7.3.3 Interacción táctil en interfaces gráficas

Las interfaces gráficas de usuario tradicionales no son un medio adecuado de interacción con manos virtuales como sí lo son con un dispositivo de entrada con punteros como el mouse. Sin embargo, proporcionan un lenguaje de interacción que resulta familiar a cualquier usuario que utilice por primera vez un entorno de realidad y por esa razón son aún muy utilizadas. Algunos problemas que presentan estas interfaces es que requieren reinterpretaciones de cómo se accionan los widgets. Es por eso que resulta propicio reforzar la interacción con widgets de interfaces gráficas utilizando háptica.

Módulo de interfaces gráficas de Leap Motion

Tomando de base el módulo de Leap Motion de interacciones con interfaces gráficas llamado UIInput, veremos cómo extenderlo para reforzar el feedback de interacción incorporándole cualidades hápticas. El módulo UIInput trae elementos típicos de UI, como botones, sliders y scroll views contenidos en paneles visuales flotantes que asemejan las interfaces típicas de usuario. Para poder interactuar con ellos implementa un módulo de entrada del *EventSystem* de Unity personalizado para tratar los eventos de cómo las manos interactúan con interfaces gráficas. Esto provee una base sólida sobre la cual trabajar para extender las funcionalidades.

El comportamiento de los elementos gráficos es que recrean una analogía con el comportamiento de interfaces con punteros en el cual el hover el mouse sucede cuando se presiona con los dedos, el arrastre del elemento o sliding sucede cuando se mueve la mano con el elemento presionado, y luego el accionar del elemento o click sucede cuando se levanta la mano del elemento. El problema que trae aparejado el traspaso de un medio plano con punteros a un entorno de realidad virtual es que los paneles gráficos son atravesados por las

manos. Algunos refuerzos que se emplearon para atenuar estos efectos son la incorporación de feedback con sonidos al realizar las acciones y la utilización de leves contracciones de los paneles del lado visual.

Incorporación de háptica

Una forma de incorporar háptica en interfaces gráficas es teniendo estímulos predefinidos que se desencadenen al producirse los eventos de interacción con los widgets. O aproximación sería que los elementos gráficos reaccionen al tacto como si se tratase de un objeto físico de interacción. En este caso presentamos una solución basada en esta última aproximación la cual funciona en analogía con el resto de las soluciones.

La solución consiste en dotar los elementos de interfaz con objetos colisionables de modo que al tocarlos se produzca la interacción háptica. De este modo la solución se construye de manera similar a los objetos plegables de contacto prolongado vistos anteriormente. La diferencia con estos elementos antes vistos es que los elementos de interfaz no tienen un mesh asociado así que hay que inventar como sería el colisionador, y que la forma de plegarse tiene que simular una presión ejercida sobre el mismo eje del elemento visual.

Los pasos para construir el colisionador consisten en generar un objeto que esté contenido dentro del widget y que simule la parte de la simulación háptica. El objeto debe estar dotado con un *BoxCollider* que cubra el ancho y alto del widget y con un grosor mínimo. Agregar un Rigidbody para la interacción física con propiedades de drag altas para que el objeto se corra lo justo y necesario sobre el movimiento que le impulse la mano. En este componente es donde se configura la fijación del eje de interacción, el cual se realiza cancelando la rotación del objeto. Luego se utiliza el componente *HapticMaterialDepth* para describir el material táctil con que reacciona la mano al entrar

en contacto con el colisionador. Por último, se agrega el helper *RetractableObjet* con un valor alto de velocidad de retracción para retraer el objeto y volverlo a su posición inicial. Este proceso se repite en cada widget del panel de la interfaz gráfica. En las utilidades del proyecto se provee un prefab llamado *RetractableUI* el cual se puede usar con este fin, teniendo solo que configurar el tamaño del *Collider* para adaptarlo a la forma de cada widget.

7.3.4 Diseño de estímulos táctiles predefinidos

Hay situaciones en las cuales el estímulo háptico que se quiere reproducir no puede ser expresado en términos de contactos físicos con materiales o simplemente se quiere tener un control más fino del estímulo a recrear. En estos casos la herramienta contempla la utilización de animaciones que describan el estímulo conjuntamente con componentes que detecten y disparen estos efectos.

La animación de los estímulos se realiza sobre el motor de animaciones provisto nativamente por Unity. Las animaciones de Unity, llamadas internamente AnimationClip, consisten en una línea de tiempo donde por cada frame se describe el valor de las propiedades de los objetos animados. Las líneas de tiempo pueden ser manipuladas utilizando un esquema de dopesheet o basado en curvas. Con el esquema basado en curvas, los estímulos táctiles pueden ser editados en correspondencia con la metodología prevalente en herramientas de autoría háptica por representación directa evaluadas previamente.

Para demostrar el diseño de estímulos en la herramienta, implementaremos un caso de uso en el cual el usuario al tocar un objeto desencadene una explosión que produzca chispazos en las manos por un breve periodo de tiempo. La interacción háptica que hay que reproducir es, primero cuando se produce el contacto con el objeto, el cual

se comporta igual que los casos vistos anteriormente, y luego la recreación de un estímulo que simule que se están produciendo chispazos en las manos del usuario.

Como primer realizaremos el diseño del estímulo. Este se realiza mediante la manipulación directa de los valores de los *Actuators* de los huesos dentro del panel de animación de Unity. En el caso de que se utilice el modelo gráfico de manos provisto por la herramienta será posible visualizar el estímulo en el editor al mismo tiempo que se diseña, dado el coloreado de los huesos de las manos producido por el modelo gráfico. La animación se referencia en la maquina de estados del *AnimatorController* junto con las transiciones que se utilicen para configurar el comienzo y el fin de la animación. Es importante notar que las animaciones, que representan el estímulo háptico, quedan asociadas en este caso a las manos, a diferencia de los otros casos donde la representación de los materiales/estímulos se asociaba a los objetos de interacción. Esto es una salvedad en el esquema propuesto en la herramienta para sacar provecho de la manipulación directa de los valores de los *Actuators* cuando se utiliza el motor de animaciones de Unity.

El componente de detección de las manos provisto por la herramienta contempla un tipo de interacción háptica en la cual, cuando la mano entra en contacto con un objeto compuesto por un *HapticEffectTrigger,* dispare en la mano la ejecución de un script cuyo nombre corresponda con el nombre del efecto asignado. Este script es el encargado de iniciar y terminar la animación del estímulo háptico conjuntamente con otros efectos visuales o sonoros que se desencadenen por motivo de la interacción.

En nuestro caso de uso describiremos un objeto llamado Bomb el cual contenga los elementos típicos que lo hacen interactuables, es decir un *Collider* y un *Rigidbody*. Además, en vez de usar el componente *HapticMaterialDepth,* vamos a usar un componente llamado *HapticEffectTrigger* el cual designe un efecto a disparar llamado “FireHan-

dEffect". El nombre de este efecto, tal como describimos anteriormente, producirá la ejecución de un script de mismo nombre localizado en el modelo físico de las manos. Este script al ejecutarse inicia la animación del estímulo y al mismo tiempo lanza efectos visuales de chispas y sonidos correspondientes.

En los ejemplos provistos dentro de la herramienta se encuentra un prefab con el objeto *Bomb* aquí descripto y el ejemplo de un posible efecto háptico de chispas en la animación *FlareEffect*.

8 CONCLUSIONES

En esta sección presentamos un resumen del trabajo de investigación realizado, resaltando las principales contribuciones, para luego dar una perspectiva de los trabajos futuros que quedan pendientes de ser completados.

8.1 RESUMEN Y CONTRIBUCIONES

En el presente trabajo nos propusimos investigar cómo integrar herramientas de diseño háptico en entornos de desarrollo de realidad virtual de modo de conformar un modelo experimental donde diseñadores pudieran diseñar interacciones y estímulos de forma simple e intuitiva. Bajo la hipótesis de que el contexto actual amerita la mejora de las herramientas de diseño en háptica y que las plataformas más adecuadas para su funcionamiento son los motores de videojuegos, donde se permite la coexistencia con el resto de las tecnologías de desarrollo en realidad virtual.

Para el desarrollo de la herramienta de autoría aquí presentada se seleccionó un conjunto de tecnologías de realidad virtual sobre las cuales trabajar en base a las disponibles en ese momento. Con el objetivo final de construir una herramienta de autoría háptica de carácter experimental, fijamos un alcance limitado a ciertas premisas que buscamos satisfacer. Las cuales trataban sobre construir una herramienta aprovechando todo el potencial de las tecnologías disponibles y que permitiera construir entornos virtuales con háptica de manera rápida y sencilla. Fueron analizados trabajos de las últimas dos décadas que trataban de la construcción y aspectos a tener en cuenta en el desarrollo de herramientas de autoría háptica, los cuales fueron

recopilados en el capítulo de Trabajos Relacionados y sobre los cuales se extrajeron importantes requerimientos. La herramienta fue desarrollada en ciclos iterativos en los cuales se implementaron funcionalidades utilizando diferentes aproximaciones; las cuales fueron probadas, descartadas y mejoradas, hasta que alcanzaron suficiente proximidad con la modalidad que habíamos propuesto. Por último, realizamos y documentamos un conjunto de pruebas de concepto en las cuales se demuestran las características de la herramienta y se da un vistazo a algunas posibles creaciones que se pueden realizar a partir de ella.

Con las pruebas realizadas sobre la herramienta pudimos comprobar la hipótesis planteada bajo una serie de contribuciones manifiestas al diseño háptico. El acople del diseño háptico al conjunto de herramientas para desarrollo de entornos virtuales es uno de los puntos más destacables en los beneficios obtenidos. Por ejemplo, en el hecho de acceder a la posibilidad de diseñar al mismo tiempo el comportamiento de los objetos, junto con la visualización gráfica que representa y en simultaneo diseñar sus propiedades hápticas. Probamos que esta integración permite acelerar los tiempos y aumenta la capacidad creativa de los diseñadores. También destacamos la facilidad obtenida en el acceso a una herramienta de autoría háptica de estas características y en la reducción de los conocimientos previos necesarios para utilizarla, ya que está construido sobre un entorno familiar y conocido por muchos usuarios.

8.2 Trabajos futuros

Según el alcance que fijamos inicialmente, el objetivo principal que nos condujo fue el de la integración de una herramienta de autoría en un entorno de desarrollo de realidad virtual. Para esto, basamos

nuestro enfoque en lograr la integración de buenas prácticas de diseño háptico reunidas de investigaciones previas con un flujo de trabajo acorde al editor Unity y los procesos de desarrollo de mundos virtuales. Es por eso que algunas problemáticas del diseño de estímulos hápticos que consideramos secundarias no fueron abordadas para poder mantener el alcance del proyecto. De esta manera muchos puntos interesantes relacionados con la optimización de la herramienta y la mejora del flujo de trabajo quedaron pendientes para investigar. A continuación, enumeramos algunos de ellos.

MODO DE DISEÑO POR CARACTERÍSTICAS FÍSICAS

Un paradigma alternativo y que podría complementar nuestra herramienta es el modo de diseño basado en las características físicas de los objetos. En este modo, la especificación de los estímulos se realiza mediante la descripción de las propiedades hápticas del objeto en lugar de describir el estímulo con representaciones directas como la curva. El funcionamiento de una simulación háptica de este tipo consiste en que la herramienta genere la representación del estímulo dinámicamente según las propiedades hápticas del objeto y las características de la interacción que esté ocurriendo en ese momento. Un ejemplo de herramienta con este modo de diseño es HAMLAT [39] donde los objetos virtuales son descriptos con propiedades hápticas como rigidez, factor de amortiguación y fricción. Luego el rendering háptico es exportado para ser utilizado en los dispositivos de force feedback de la serie PAHNTOM.

En nuestra herramienta realizamos una aproximación a este paradigma en la utilización de un descriptor físico en la implementación del componente *HapticMaterialDepth*. El cual está relacionado a las características de rigidez y amortiguación del objeto. En este componente, utilizamos una curva para representar la fuerza emitida en función del grado de penetración. Del mismo modo, esto podría ser

extendido con una herramienta para describir la textura de los materiales y que haga alusión a la rugosidad y suavidad de los objetos. Sin embargo, una herramienta que describa en su conjunto las propiedades hápticas mencionadas requiere el rediseño de la aproximación empleada ya que debe considerarse que varias propiedades hápticas estén activas en un mismo momento y como se articulan todas ellas.

Consolidación del flujo de trabajo y escalabilidad

Desde el punto de vista del diseño de la herramienta, nuestro enfoque de implementación cambió drásticamente en sucesivas iteraciones. De nuestro enfoque inicial que iba a ser una construcción desde cero de interfaces de autoría háptica integrada al motor de videojuegos, cambiamos hacía una aproximación de diseño basada en la extensión y reutilización de las herramientas propias de Unity. Esto sucedió debido a que Unity ya tiene consolidadas una serie de metodologías y mejores prácticas. De este modo, al trabajar en la simplificación de las extensiones sobre el editor, el proceso confluye hacía la reutilización de determinadas abstracciones y procedimientos que son efectivos y resultan familiares a sus usuarios. Este proceso de mejora y confluencia de estilos de diseño creemos que se puede mejorar aún más.

Por otro lado, garantizar la escalabilidad del proyecto requiere desacoplar la herramienta del Core Assets de Leap Motion. Ya sea para que pueda ser utilizada con diferentes dispositivos de reconocimiento de gestos como así también para el soporte de futuras versiones de los módulos de Leap Motion, los cuales son al día de hoy provistos en carácter experimental.

Diseño de estímulos por intensidad percibida

En el diseño del estímulo suele presentarse como inconveniente lidiar con los retrasos de aceleración y desaceleración de los motores. Esto ocurre cuando se produce una divergencia entre el estímulo diseñado y la intensidad percibida por el usuario. Ya que las curvas representan solo la intensidad de la señal en un instante dado enviada a los motores vibrotáctiles, sin embargo, la fuerza producida varía según las características de cada motor. En general, esto se mitiga iterando el diseño del estímulo con sucesivas pruebas y aplicando correcciones. Por otro lado, el ajuste de la intensidad global de las señales puede mitigarse incluyendo en la herramienta una función de ajuste de amplitud, práctica que se ha visto aplicada desde los primeros editores hápticos [13]. Sin embargo, esto no soluciona el hecho de que cada motor tiene distintos retrasos y niveles de intensidad, produciendo un diseño de estímulos ligado a la especificación de cada motor y no a la intensidad percibida por los usuarios.

Una posible solución sería proveer un diseño de estímulos donde las curvas representen la intensidad percibida por el usuario en vez de la fuerza emitida en un momento determinado por el motor. Es importante notar que implementar una solución a este problema podría implicar tareas tales como el estudio de las variantes en la sensibilidad humana al tacto y la consideración de los retrasos de distintas tecnologías de motores vibrotáctiles. Un modo de diseño de estas características es el llamado *Perceptually Transparent Rendering* presentado en [16] el cual minimiza las distorsiones percibidas por el usuario del efecto vibrotáctil usando una función de magnitud psicofísica.

9 BIBLIOGRAFÍA

[1] R. L. K. Lederman, S. J., "Haptic perception: A tutorial," *Atten. Percept. Psychophys.*, vol. 71, no. 3, pp. 481–489, 2009.

[2] N. M. Richardson, "One Giant Leap for Mankind," 2013. [Online]. Available: https://www.inc.com/30under30/nicole-marie-richardson/leap-motion-david-holz-michael-buckwald-2013.html.

[3] A. Colgan, "Getting Started with Unity and the Oculus Rift," *Leap Motion Blog*, 2014.

[4] A. Colgan, "HTC Vive FAQ: What You Need to Know About Leap Motion + SteamVR," *Leap Motion Blog*, 2016.

[5] Y. Pulijala, M. Ma, and A. Ayoub, "VR Surgery: Interactive Virtual Reality Application for Training Oral and Maxillofacial Surgeons using Oculus Rift and Leap Motion," *Serious Games Edutainment Appl.*, vol. Volume II, p. pp 187-202, 2017.

[6] T. Hilfert and M. König, "Low-cost virtual reality environment for engineering and construction," *Vis. Eng.*, 2016.

[7] J. Fillwalk, "ChromaChord: A virtual musical instrument," *3D User Interfaces (3DUI), 2015 IEEE Symp.*, 2015.

[8] H. Vincent and K. Maclean, "Do it yourself haptics: part I," *IEEE Robot. Autom. Mag.*, vol. 14, no. Dec. 2007, 2007.

[9] J. San Martin and G. Trivino, "A study of the Manipulability of the PHANToM OMNI Haptic Interface.," *Proc. Third Work. Virtual Real. Interact. Phys. Simulations*, 2006.

[10] K. Salisbury, F. Conti, and F. Barbagli, "Haptic rendering: Introductory concepts," *IEEE Comput. Graph. Appl.*, vol. 24, no. 2, pp. 24–32, 2004.

[11] J. M. Muñoz, "A Vibrotactile Prototyping Toolkit for Virtual Reality and Videogames," pp. 28–41, 2013.

[12] W. W. Gaver, "Auditory Icons: Using Sound in Computer Interfaces," *Appl. Psychol.*, no. May 2015, pp. 37–41, 1986.

[13] M. J. Enriquez and K. E. MacLean, "The hapticon editor: A tool in support of haptic communication research," *Proc. - 11th*

Symp. Haptic Interfaces Virtual Environ. Teleoperator Syst. HAPTICS 2003, pp. 356–362, 2003.

[14] C. Swindells, E. Maksakov, K. E. MacLean, and V. Chung, "The role of prototyping tools for haptic behavior design," *Proc. - IEEE Virtual Real.*, vol. 2006, p. 90, 2006.

[15] G. S. Lee and B. Hannaford, "Anisotropies of touch in haptic icon exploration," *Proc. 2003 IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS 2003) (Cat. No.03CH37453)*, vol. 3, no. October, pp. 2713–2717, 2003.

[16] J. Ryu, "Perception-based Vibration Rendering in Mobile Device," 2010.

[17] J. Lee, "Vibrotactile Score for Designing Vibrotactile Patterns," *Div. Electr. Comput. Eng. (Computer Sci. Eng.*, 2009.

[18] M. Jonas, "Tactile Editor: A Prototyping Tool to Design and Test Vibrotactile Patterns," 2008.

[19] S. Panëels, M. Anastassova, and L. Brunet, "TactiPEd: Easy prototyping of tactile patterns," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 8118 LNCS, no. PART 2, pp. 228–245, 2013.

[20] K. Hong, J. Lee, and S. Choi, "Demonstration-based vibrotactile pattern authoring," *TEI 2013 - Proc. 7th Int. Conf. Tangible, Embed. Embodied Interact.*, pp. 219–222, 2013.

[21] D. Cuartielles, a. Göransson, T. Olsson, and S. Stenslie, "Developing Visual Editors for High-Resolution Haptic Patterns," *Seventh Int. Work. Haptic Audio Interact. Des.*, pp. 42–44, 2012.

[22] "Immersion Announces MOTIV Development Platform for Android," 2011.

[23] K. Minamizawa, Y. Kakehi, M. Nakatani, S. Mihara, and S. Tachi, "TECHTILE toolkit: a prototyping tool for design and education of haptic media," *Proc. 2012 Virtual Real. Int. Conf. - VRIC '12*, p. 1, 2012.

[24] M. Eid, S. Andrews, A. Alamri, and A. El Saddik, "HAMLAT: A HAML-based authoring tool for Haptic application development," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 5024 LNCS, pp. 857–866, 2008.

[25] a C. Fyans and G. McAllister, "Creating Games with Feeling," 2006.

[26] M. Poyade, M. Kargas, and V. Portela, "Haptic Plug-in for Unity," *Digit. Des. Stud. (DDS), Glas. Sch. Art, Glas. United Kingdom*, pp. 1–23, 2014.

[27] K. E. MacLean and V. Hayward, "Do It Yourself Haptics: Part II [Tutorial]," *IEEE Robot. Autom. Mag.*, vol. 15, no. 1, pp. 104–119, 2008.

[28] C. Mousette, *Simple Haptics*, no. October. 2012.

[29] J. Rix, S. Haas, and J. Teixeira, *Virtual Prototyping*. Springer US, 1995.

[30] V. R. D. C. GDC, "VR / AR Innovation Report," no. August, pp. 1–11, 2016.

[31] "Unity - Fast Facts," 2016.

[32] G. C. Burdea, "Keynote Address: Haptic Feedback for Virtual Reality," *Proc. Int. Work. Virtual prototyping, Laval, Fr.*, no. May, pp. 87–96, 1999.

[33] G. C. Burdea and P. Coiffet, "Virtual Reality Technology," *John Wiley Sons*, vol. 1, 2003.

[34] S. T. Shogo Okamoto, Masashi Konyo, Satoshi Saga, "Detectability and Perceptual Consequences of Delayed Feedback in a Vibrotactile Texture Display," *IEEE Trans. Haptics*, 2009.

[35] "Games by the Numbers Q2 2016," 2016.

[36] "What's new in Unity 5.1," 2015.

[37] "Unity Manual - VR Overview," 2016.

[38] M. Cohn, *Succeeding with Agile Software Development Using Scrum*. Addison-Wesley, 2010.

[39] S. Andrews, M. Eid, A. Alamri, and A. El Saddik, "Extending Blender: Development of a haptic authoring tool," *HAVE 2007 - 6th IEEE Int. Work. Haptic, Audio Vis. Environ. Games, Proc.*, no. October, pp. 44–49, 2007.